UNIVERSIDADE NOVA DE LISBOA
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica

Geração Automática de Controladores em FPGA Integrando Animação Gráfica

Por:
Filipe de Carvalho Moutinho

Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa para obtenção do grau de Mestre em Engenharia Electrotécnica e de Computadores

Orientador: Prof. Doutor Luís Gomes

Lisboa
2008

Sumário

Este trabalho tem o objectivo de gerar automaticamente o código de implementação para controladores de processos, integrando a actualização do sinóptico associado. Os controladores considerados vão ser implementados em plataformas reconfiguráveis baseadas em FPGA (*Field Programmable Gate Arrays*), integrando interface gráfica dedicada (*LCD – Liquid Crystal Display*) bem como interfaces de entrada/saída permitindo a sua interligação física ao processo a controlar.

Para tal, desenvolveu-se uma ferramenta denominada *Animator4FPGA*, que gera código VHDL, com a função de criar e animar o sinóptico apresentado no *LCD*, interligar o sinóptico ao controlador e associar as entradas/saídas do modelo do sistema às entradas/saídas da plataforma de implementação.

O sistema é modelado através de redes de Petri (RdP) IOPT, editadas na ferramenta *Snoopy-IOPT*, que tem também a função de descrever a RdP IOPT no formato PNML.

O *Animator4FPGA* recebe como entrada um conjunto de regras que associam as características do modelo comportamental do sistema às características do sinóptico. Este conjunto de regras é definido utilizando a aplicação *Animator*.

O código do controlador é gerado pela ferramenta *PNML2VHDL*, que gera código VHDL a partir do PNML da RdP IOPT. As ferramentas *Snoopy-IOPT*, *Animator* e *PNML2VHDL* foram realizadas no âmbito de outras dissertações de Mestrado.

Abstract

This work aims to automatically generate process controller's implementation code that integrates updating synoptic. These controllers will be implemented on reconfigurable platforms based on FPGA (*Field Programmable Gate Arrays*), incorporating dedicated graphical interfaces (*LCD – Liquid Crystal Display*) and input and output interfaces allowing its physical connection to the process under control.

A tool called *Animator4FPGA was developed*, allowing the generation of VHDL code, with the goal of creating and animating the synoptic of the process under control, linking the synoptic to the controller, and mapping system model inputs and outputs to implementation platform inputs and outputs.

The system is modeled using IOPT Petri Nets (PN), edited with the *Snoopy-IOPT* tool, which also has the function of generate PNML representation of the IOPT PN model.

The *Animator4FPGA* receives as input a set of rules that associates the characteristics of the behavioral model of the system to the characteristics of the associated synoptic. This set of rules is defined using the *Animator* tool.

The controller code is generated by the *Pnml2Vhdl* tool, which generates VHDL code from the PNML representation of the IOPT net model. The tools *Snoopy-IOPT*, *Animator* and *PNML2VHDL* were developed under other MSc degree works.

Simbologia e Notações

∩	Intersecção de conjuntos
N	Números naturais
∈	Pertença a um conjunto
∀	Quantificador universal
U	União de conjuntos
BMP	Windows bitmap
BRITNeY	Basic Real-Time Interactive Tool for Net-based animation
FPGA	Field Programmable Gate Array
IOPT	Input-Output Place-Transition
PNML	Petri Net Markup Language
RdP	Rede de Petri
Rx	Pin de recepção de dados da linha série
SCADA	Supervisory, Control and Data Acquisition
VHDL	VHSIC (Very High Speed Integrated Circuits) Hardware Description Language

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1 – Introdução

1.1. Enquadramento

A permanente evolução ao nível do hardware permite que se desenvolvam sistemas electrónicos cada vez mais complexos, respondendo a um número cada vez maior de requisitos.

Este constante aumento da complexidade dos sistemas faz com que as exigências ao nível da sua modelação sejam cada vez maiores. Ter um modelo que descreve adequadamente o sistema é uma mais valia quer para o seu desenvolvimento, quer para a sua documentação.

Adicionalmente, se for possível gerar código automaticamente a partir do modelo, reduz-se consideravelmente o tempo gasto no desenvolvimento e evitam-se os comuns erros de implementação manual.

O projecto FORDESIGN [FORDESIGN, 2007] vai de encontro a estas questões de modelação e de geração automática de código, e é neste projecto que se inserem os trabalhos associados a esta dissertação com o título de Geração Automática de Controladores em FPGA Integrando Animação Gráfica.

O autor considera que esta dissertação poderá constituir uma mais valia, dada a falta de ferramentas de geração automática de código que cumpram os objectivos propostos para o trabalho, que se resumem na secção seguinte.

1.2. Objectivos

Desenvolver um ambiente de geração automática de arquitecturas descritas em VHDL, que irão ser usadas na configuração de plataformas reconfiguráveis baseadas em FPGA, integrando interfaces gráficas dedicadas (LCD) e interfaces de entrada/saída.

O objectivo é o de utilizar uma plataforma reconfigurável como controlador de processos (com funções semelhantes a autómatos programáveis) integrando capacidades de actualização do sinóptico do processo sob controlo (apresentado num LCD acoplado, com funções semelhantes a um sistema SCADA).

Pretende-se igualmente explorar a associação entre características relevantes do modelo comportamental do sistema e as características do sinóptico associado, bem como a animação desse sinóptico. Esta associação será expressa através de regras, que serão avaliadas e executadas pela aplicação gerada.

As redes de Petri foram utilizadas como formalismo para modelar o comportamento do sistema, permitindo explorar a associação das suas características estáticas (marcação da rede) e das suas características dinâmicas (disparo de transições) às características gráficas do sinóptico pretendido.

A ferramenta desenvolvida foi utilizada em conjunção com outras ferramentas desenvolvidas no âmbito do projecto FORDESIGN (no qual este trabalho também se insere) nomeadamente as ferramentas de edição e de geração automática de código VHDL, permitindo reforçar as características de geração automática de código a partir de modelos comportamentais.

Para validar o protótipo obtido foram utilizados casos específicos de sistemas embutidos modelados com redes de Petri.

1.3. Estrutura da Dissertação

Este documento está organizado da seguinte forma: no segundo capítulo, descrevem-se os conceitos teóricos que fundamentam o trabalho e apresentam-se os resultados da pesquisa efectuada sobre os trabalhos realizados na área; no terceiro capítulo, é apresentado o fluxo de desenvolvimento dos controladores; no quarto capítulo, faz-se uma caracterização da plataforma de implementação; no quinto capítulo, apresentam-se de forma detalhada as soluções encontradas para as implementações do controlador integrando interface gráfica, do gravador de imagens na EEPROM e do Animator4FPGA; no sexto capítulo, apresenta-se um exemplo de aplicação; no sétimo e último capítulo, apresentam-se as conclusões e sugestões para trabalhos futuros; e em anexo, um breve manual de utilizador.

Capítulo 2 – Fundamentação Teórica

Neste capítulo, descreve-se o formalismo de modelação que serve de base ao trabalho (as redes de Petri IOPT); a tecnologia PNML, que vai ser usada para representar as redes de Petri IOPT que modelam o sistema; e os trabalhos relacionados, resultado da pesquisa efectuada.

2.1. Modelação de Sistemas através de Redes de Petri IOPT

Definidas em [Gomes et al., 2007b][Pais, Barros & Gomes, 2005], são uma extensão às classes de redes de Petri lugar-transição [Reisig, 1985], adicionando as seguintes características:

- As entradas (sinais e eventos) do controlador podem ser associadas às transições;
- As saídas (sinais e eventos) do controlador podem ser associadas às transições e aos lugares;
- É possível definir prioridades nas transições;
- Os lugares têm um atributo "bound" (limite de marcas);
- Dois tipos de valores para os sinais de entrada e de saída;
- Uma especificação explicita para um conjunto de transições com conflitos;
- Uma especificação explicita para um conjunto de transições síncronas;
- Arcos de teste.

Definição 1 (Interface do sistema)

Os sinais e eventos de entrada e de saída servem de interface entre o sistema e o controlador. A interface do sistema controlado (*ICS*) com a rede de Petri IOPT é

$$ICS = (IS, IE, OS, OE)$$

Onde:

- *IS* é um conjunto finito de sinais de entrada
- *IE* é um conjunto finito de eventos de entrada
- *OS* é um conjunto finito de sinais de saída
- *OE* é um conjunto finito de eventos de saída
- $IS \cap IE \cap OS \cap OE = 0$

Definição 2 (Estado de entrada do sistema)

Considerando um interface do sistema com as características da definição 1, o estado de entrada do sistema (*SIS*) é definido por

$$SIS = (ISB, IEB)$$

Onde:

- *ISB* é um conjunto finito de estados dos sinais de entrada, $ISB = IS \times N_0$
- *IEB* é um conjunto finito de estados dos eventos de entrada, $IEB = IE \times B$

Definição 3 (Redes IOPT)

A redes IOPT são descritas por uma sintaxe concreta, o que permite a especificação de expressões algébricas, variáveis, e funções para a especificação de guardas nas transições e condições nas acções de entrada associadas aos lugares.

Dado um controlador com um interface que tenha as características da definição 2, uma rede IOPT é um conjunto

$$N = (P, T, A, TA, M, weight, weightTest, priority, isg, ie, oe, oce)$$

Onde:

- *P* é um conjunto finito de lugares
- *T* é um conjunto finito de transições (disjunto de *P*)
- *A* é um conjunto de arcos, tal que $A \subseteq ((P \times T) \cup (T \times P))$
- *TA* é um conjunto de arcos de teste, tal que $TA \subseteq (P \times T)$
- M é a função de marcação: $M : P \to N_0$
- Peso dos arcos, $weight : A \to N_0$
- Peso dos arcos de teste, $weightTest : TA \to N_0$
- A prioridade ($priority$) é uma função parcial que aplica transições a inteiros não negativos, $priority : T \to N_0$
- isg é uma função parcial de guarda de sinal de entrada que aplica transições a expressões booleanas (onde todas as variáveis são sinais de entrada): $isg : T \to BE$, onde $\forall eb \in isg(T), Var(eb) \subseteq IS$. *BE* é o conjunto de expressões booleanas. $Var(E)$ retorna o conjunto de variáveis numa dada expressão E
- ie é uma função parcial de eventos de entrada que aplica transições a eventos de entrada: $ee : T \to IE$
- oe é uma função parcial de eventos de saída que aplica transições a eventos de saída: $ee : T \to OE$
- osc é uma função para um sinal de saída, de lugares para conjuntos de regras: $osc : P \to P(RULES)$, onde $RULES \subseteq (BES \times OS \times N_0)$, $BES \subseteq BE$ e

$\forall e \in BES, Var(e) \subseteq ML$, *ML* é o conjunto de identificadores para cada marcação de lugar, após um estado de execução cada marcação de lugar tem associado um identificador, que é usado quando o código gerado é executado

As transições têm sempre associadas prioridades e eventos de entrada e de saída. Podem ter também guardas, que são função de sinais de entrada externos. Os sinais de saída são alterados com base no disparo das transições ou com base na marcação dos lugares.

A semântica de execução das redes IOPT diz que a transição dispara quando a transição está habilitada e a condição de entrada externa é verdadeira (o evento de entrada e o guarda do sinal de entrada são verdadeiros).

As redes IOPT são redes sincronizadas, o que implica que a evolução da rede só é possível em determinados instantes de tempo, definidos por um relógio global externo.

2.2. PNML

O PNML (Petri Net Markup Language) [PNML, 2004] foi criado para representar em formato de ficheiro as redes de Petri da versão Java do Petri Net Kernel [Petri Net Kernel, 2002]. O PNML é baseado em XML, tal como outros formatos desenvolvidos para representar redes de Petri. De referir que não existe nenhum formato standard para representar redes de Petri.

Uma característica do PNML é que define as características comuns a todas as classes de redes de Petri num ficheiro PNML, e as características particulares num ficheiro PNTD (Petri Net Type Definition).

Existe um documento de convenções, que contém as características especificas de cada classe de redes de Petri. Desta forma pretende-se garantir o intercâmbio de informação entre diferentes ferramentas baseadas em redes de Petri.

2.3. Trabalhos Relacionados

Esta secção é o resultado de uma pesquisa de publicações e aplicações no contexto desta dissertação. Apresentam-se as aplicações BRITNeY Suite, Animator, Synoptic como exemplos de ambientes de desenvolvimentos com semelhanças com o ambiente desenvolvido no âmbito dos trabalhos desta dissertação, bem como uma plataforma reconfigurável disponível comercialmente que apresenta algumas das características da plataforma desenvolvida. Termina-se com umas breves conclusões.

2.3.1. BRITNeY Suite

É uma ferramenta que permite a animação de modelos formais [BRITNeY, 2008] [Westergaard, 2006] [Westergaard & Lassen, 2006] expressos em Redes de Petri Coloridas (*CPN – Colored Petri Nets*).

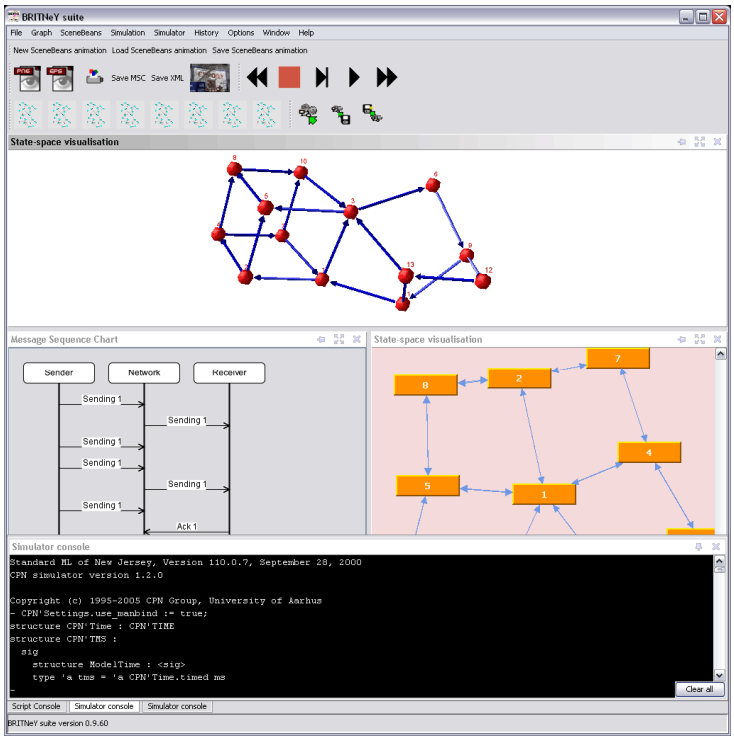


Figura 2.1. Aspecto da aplicação [BRITNeY, 2008]

Começou por ser uma simples biblioteca de animação para ser usada com CPN Tools, até que se tornou numa independente e completa plataforma de animação para ser usada com CPN Tools e não só.

Dado que utiliza um protocolo de comunicação aberto, o protocolo XML-RPC, que é um protocolo simples de RCP (Remote Procedure Call) codificado em XML, permite animar outras aplicações para além das CPN Tools. Este protocolo permite também que as aplicações que se pretendem animar estejam a ser executadas remotamente.

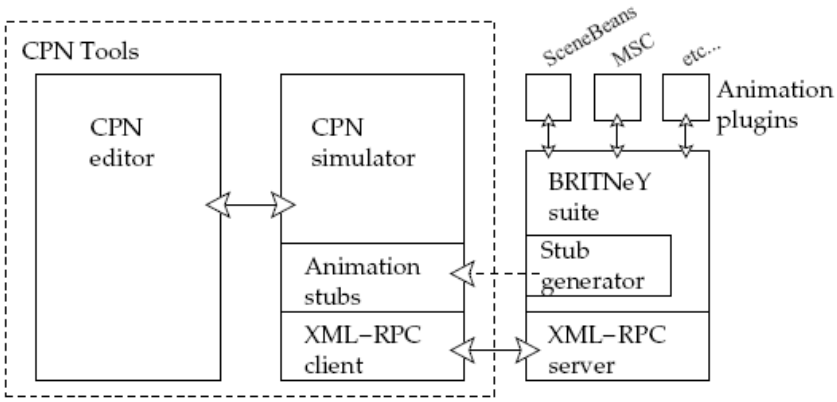


Figura 2.2 - Arquitectura do BRITNeY Suite [Westergaard & Lassen, 2006]

CPN Tools é uma ferramenta para edição, simulação e análise de redes de Petri coloridas, que são redes de Petri de alto nível.

2.3.2. *Animator* e *Synoptic*

Estas aplicações foram desenvolvidas no âmbito de uma dissertação de mestrado [Lourenço, 2008] e estão enquadradas no projecto FORDESIGN. São "vocacionadas para a geração automática e para a animação de sinópticos associados a aplicações de controlo de sistemas, modelados através de redes de Petri.".

Figura 2.3. Aspecto da ferramenta *Animator*

Estas duas aplicações estão dependentes de outras também desenvolvidas no âmbito do projecto FORDESIGN, o Snoopy-IOPT e o PNML2C.

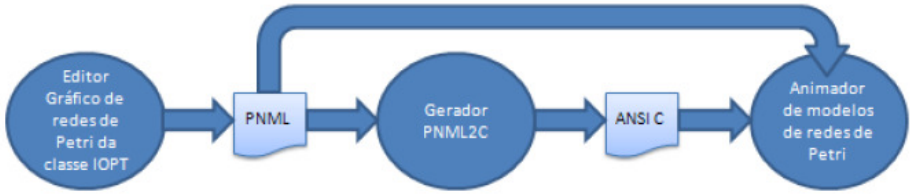


Figura 2.4. Ferramentas desenvolvidas no âmbito do projecto FORDESIGN utilizadas no *Animator* [Lourenço, 2008]

O Animator permite a definição das características gráficas do sinóptico associando-as às características estáticas (estado de marcação e estado dos sinais de entrada e de saída) e dinâmicas (disparo das transições e ocorrência de eventos) da rede de Petri.

Figura 2.5. Relação entre as características do modelo e os atributos gráficos do sinóptico [Gomes et al, 2007a]

Esta associação entre as características do modelo e as características estáticas do sinóptico tem a denominação de Regras, na figura 2.6. pode ver-se como são caracterizadas as regras.

IF	Antecedente	THEN	Consequente
	Características da rede de Petri		Características gráficas - GUI

Figura 2.6. Caracterização das regras [Lourenço, 2008]

Um antecedente é uma condição, que avalia um determinado estado da RdP. Cada regra pode ter vários antecedentes, o que significa que para estar activa num determinado momento os vários antecedentes têm de ser verdadeiros nesse momento (antecedente1 e antecente2 e ...).

Um consequente descreve as alterações gráficas que vão ocorrer no sinóptico. Por exemplo: fazer aparecer ou desaparecer uma imagem, aumentar ou diminuir o *zoom* de uma imagem, movimentar uma imagem ao longo de um trajecto, com ou sem temporizador, visualização especifica de uma variável de saída.

O Synoptic é responsável pela execução do código do controlador (gerado automaticamente pela ferramenta PNML2C) que executa o modelo e pela criação e actualização do sinóptico definido no Animator.

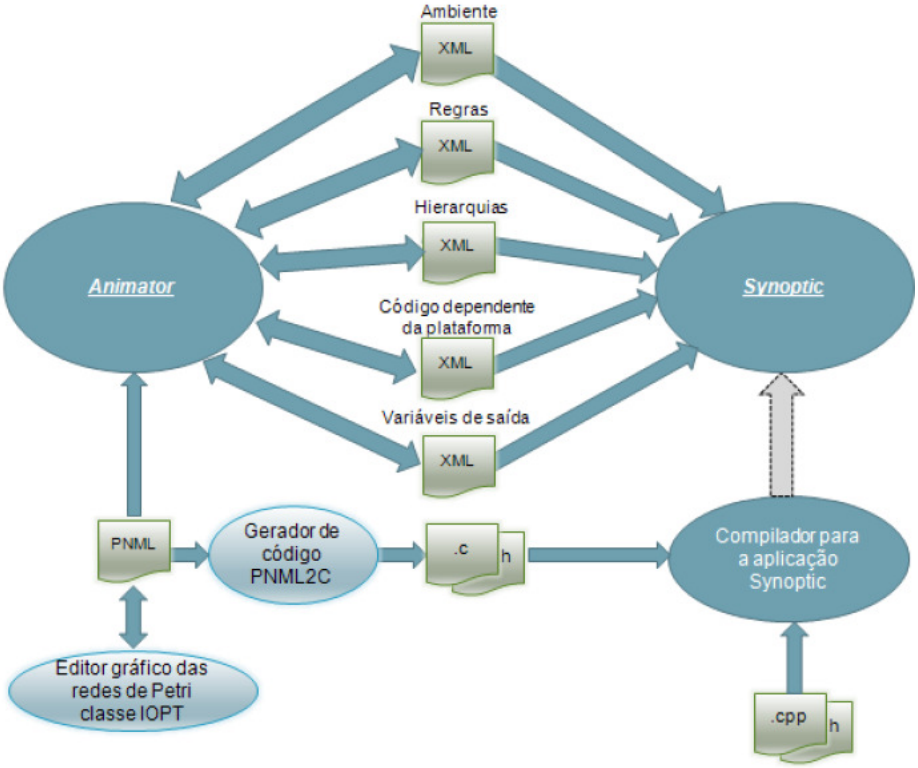


Figura 2.7. Arquitectura geral do *Animator* e do *Synoptic* [Lourenço, 2008]

Na figura 2.7., onde é apresentada a arquitectura geral, pode ver-se que o *Animator* gera 5 ficheiros XML que descrevem as características do sinóptico e as entradas do controlador. O Compilador para a aplicação *Synoptic* altera o ficheiro gerado pelo PNML2C de modo a poder ser executado pelo *Synoptic*. O *Synoptic* executa o código do controlador e cria e actualiza o sinóptico (interface com o utilizador).

2.3.3. Nios II Embedded Evaluation Kit

É uma plataforma da Altera [Nios II Kit, 2008] para sistemas embutidos. Esta plataforma contém uma FPGA Altera Cyclone III, memória SRAM, botões de pressão e interruptores, leds, fichas série, VGA, PS/2, RJ45, SD Card Socket, interface gráfico (LCD) táctil com uma resolução de

800x480, etc.. Convém realçar que a disponibilidade comercial desta plataforma apenas foi conhecida aquando do final da escrita desta dissertação.

Figura 2.8. Fotografia do Nios II Embedded Evaluation Kit [Nios II Kit, 2008]

2.3.4. Conclusões

Concluiu-se que existem poucos trabalhos relacionados, e que nenhum deles cumpre na totalidade os objectivos propostos para os trabalhos desenvolvidos nesta dissertação.

Tanto o BRITNeY Suite como o Synoptic construído pelo Animator têm a função de animar redes de Petri, mas nenhum deles em plataformas reconfiguráveis baseadas em FPGA. Não se encontrou nenhuma ferramenta de geração automática de controladores integrando animação gráfica para FPGA, sendo esta a inovação do trabalho desenvolvido nesta dissertação.

A plataforma apresentada (Nios II Embedded Evaluation Kit) é uma plataforma com algumas características semelhantes à que será usada para implementar os controladores realizados neste trabalho.

Capitulo 3 – Fluxo de Desenvolvimento

3.1. Introdução

Tal como foi dito anteriormente o objectivo é gerar automaticamente controladores em FPGA integrando animação gráfica. Neste capítulo apresenta-se o fluxo de desenvolvimento destes controladores, satisfazendo a seguinte sequência de acções:

- descrição dos requisitos do sistema a controlar através de casos de uso,
- modelação do sistema através de redes de Petri da classe IOPT,
- geração automática do código de implementação do controlador a partir do modelo expresso através de RdP IOPT,
- associação entre as características do modelo IOPT e as características gráficas do sinóptico, que definem a animação gráfica,
- geração automática do código de implementação da animação gráfica,
- implementação do código gerado anteriormente em plataformas reconfiguráveis baseadas em FPGA que integrem animação gráfica.

3.2. Arquitectura Geral

O desenvolvimento é feito utilizando várias ferramentas que interagem entre si, como se mostra na figura 3.1.. Todas elas se enquadram nos trabalhos realizados no âmbito do projecto FORDESIGN.

O Snoopy-IOPT permite a edição das RdP IOPT e a sua representação num ficheiro PNML. O PNML2VDHL gera automaticamente o código VHDL do controlador do sistema a partir do ficheiro PNML. O Animator faz a associação entre as características do modelo representado num ficheiro PNML e as características gráficas do sinóptico. O Animator4Fpga foi desenvolvido

no âmbito desta dissertação, e com base nos ficheiros gerados pelo Animator, gera automaticamente o código VHDL de implementação do sinóptico; para além disso gera também o código VHDL que interliga o código do controlador ao código do sinóptico e o ficheiro que associa as entradas e as saídas do sistema às entradas e saídas da plataforma reconfigurável baseada em FPGA.

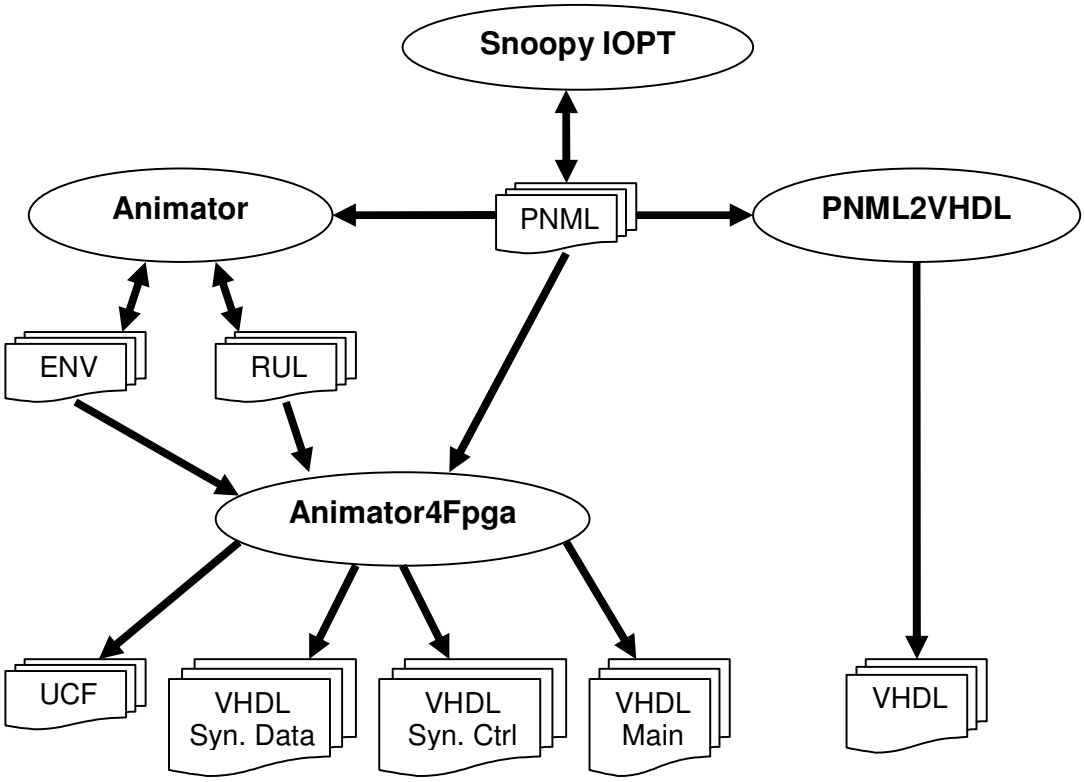


Figura 3.1. Arquitectura geral

3.3. De casos de uso a redes de Petri IOPT

Para se produzir o modelo do sistema em redes de Petri IOPT a partir de casos de uso, estão previstos 3 passos [Gomes, Costa & Meira, 2005] [Gomes et al., 2007a]:

- No primeiro passo, é feita uma inspecção de requisitos de cada caso de uso, e para cada caso de uso é produzido manualmente um modelo formal (destacam-se:

diagramas de estado, diagramas de estado concorrentes e hierárquicos, estadogramas, diagramas de sequência e classes especificas de redes de Petri);

- No segundo passo, cada um dos modelos formais produzidos anteriormente é traduzido num modelo comportamental equivalente de rede de Petri IOPT;
- No terceiro passo, são unidos todos os modelos criados no segundo passo, de modo a criar um único modelo do sistema expresso através de uma rede de Petri IOPT.

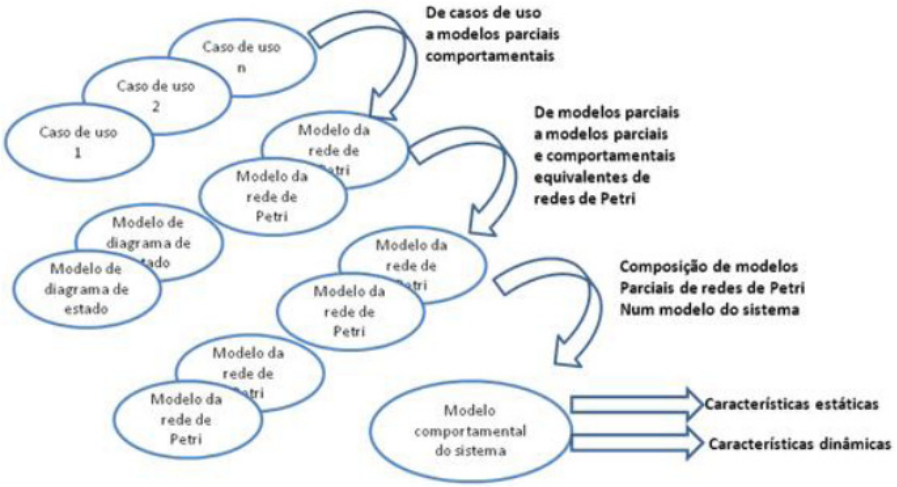


Figura 3.2. De casos de uso a redes de Petri [Lourenço, 2008]

A edição e composição de modelos é feita no editor Snoopy [Snoopy IOPT, 2008]. A união de modelos é realizada com base na adição de redes definido em [Barros & Gomes, 2004].

3.4. De redes de Petri IOPT a PNML

As redes de Petri vão ser editadas na aplicação Snoopy IOPT [SNOOPY IOPT, 2008], que é uma ferramenta de edição gráfica de redes de Petri IOPT, desenvolvida para a modelação de sistemas de automação e sistemas embutidos.

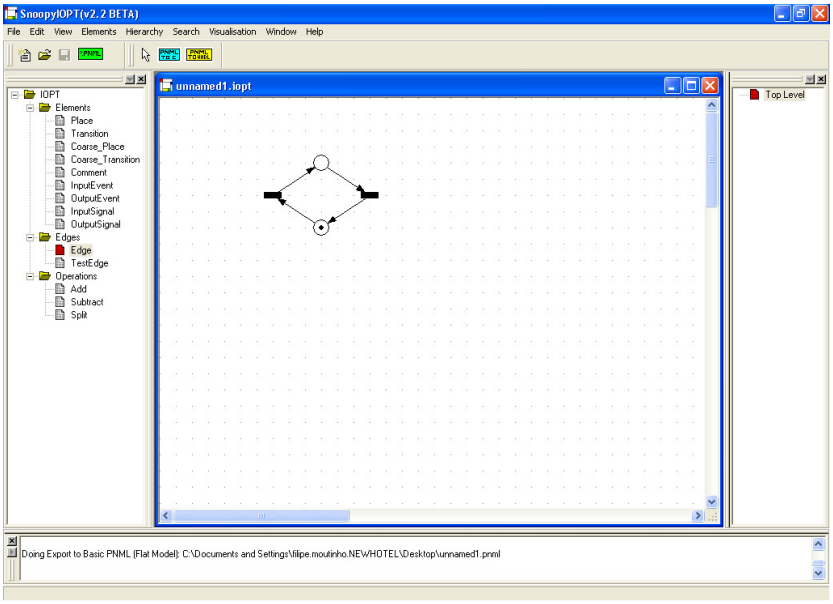


Figura 3.3. Aspecto da ferramenta Snoopy IOPT

O editor suporta especificações hierárquicas e modelares dos modelos IOPT, e a sua representação em PNML. Estas representações em PNML em conjunto com regras definidas em interfaces gráficas, podem ser usadas por um conjunto de ferramentas externas desenvolvidas no âmbito do projecto FORDESIGN [FORDESIGN, 2008]. Neste sentido, este editor funciona como ponte entre ferramentas, permitindo as seguintes tarefas:

- Modelação *top-Down* (abordagem descendente) e *bottom-up* (abordagem ascendente);
- Simulação do modelo de componentes autónomos (*token-player*);
- Representação gráfica de modelos IOPT especificados em PNML;
- Composição de modelos através de operações, nomeadamente a adição e subtracção de redes; geração de modelos através da ferramenta externa OPNML2PNML [FORDESIGN, 2007].
- Decomposição de modelos através da operação de separação; geração de modelos através da ferramenta externa SPLIT [FORDESIGN, 2007].
- Representação gráfica dos modelos compostos/ decompostos.

- Geração de código de implementação através de ferramentas externas PNML2C, PNML2VHDL.

Esta ferramenta foi desenvolvida com base no projecto “Snoopy”, gentilmente disponibilizada pelo grupo ”Data Structures and Software Dependability” da “University of Technology in Cottbus”, do departamento de “Computer Science”.

3.5. De PNML ao Controlador do Sistema

Para gerar o controlador do sistema vai-se usar a ferramenta PNML2VHDL [FORDESIGN, 2007], que gera código VHDL a partir do modelo do sistema expresso através de uma rede de Petri IOPT em formato PNML.

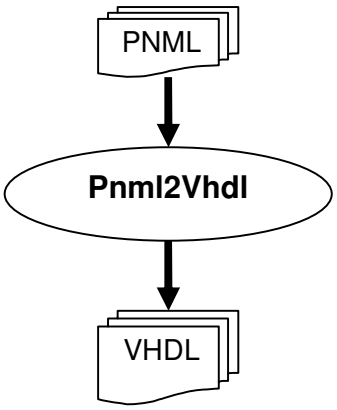


Figura 3.4. Geração de VHDL a partir do PNML

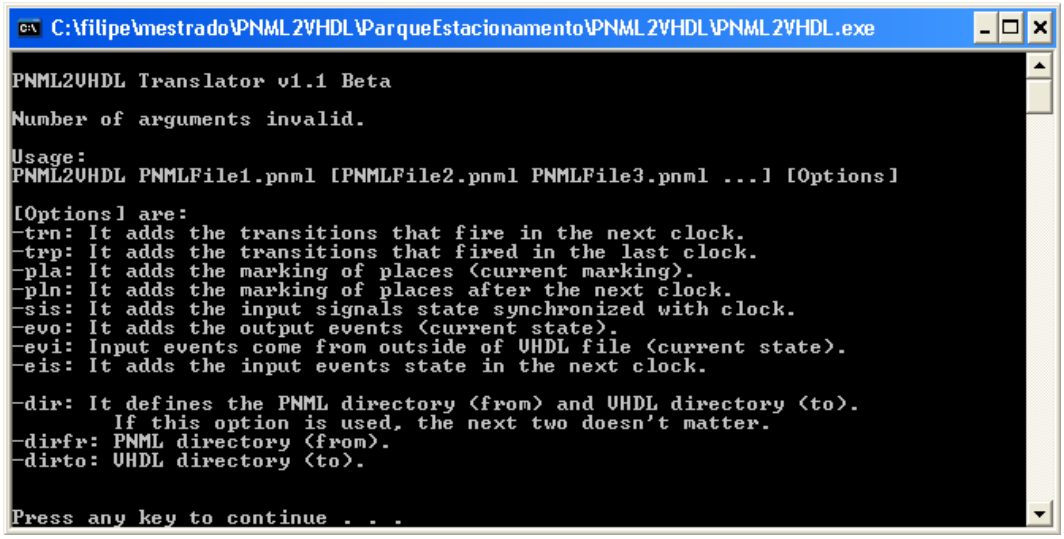


Figura 3.5. Aspecto da ferramenta PNML2VHDL

3.6. De PNML à descrição do sinóptico

Para se definir a associação entre as características gráficas do sinóptico com as características do modelo expresso através de RdP IOPT do controlador, vai-se usar a ferramenta *Animator* [Lourenço, 2008], já apresentada no capitulo 2. As funcionalidades do Animator importantes para este trabalho são:

- Criação de um fundo;
- Adição de imagens animadas ao fundo;
- Definição de um conjunto de regras que associe as características estáticas e dinâmicas da rede com os elementos gráficos constituintes do sinóptico.

As características estáticas da rede de Petri estão relacionadas com o estado da marcação dos lugares e com o estado dos sinais de entrada e de saída.

Diz-se que uma regra está activa quando o modelo se encontra no estado que está a ser avaliado pela regra.

O Animator gera vários ficheiros, onde guarda as definições feitas. Dos vários ficheiros gerados, dois são relevantes para este trabalho: o que, tem extensão ENV, e o que tem extensão RUL, ambos descritos em XML.

O ENV é o ficheiro de ambiente, onde se encontra a informação das janelas, das imagens animadas, etc.. O RUL é o ficheiro de regras, que faz a associação entre as imagens animadas e as características do modelo (RdP IOPT).

3.7. Da descrição do sinóptico à implementação

Para se gerar os ficheiros necessários para a implementação em FPGA do controlador integrando capacidades de actualização do sinóptico é necessário:

- O ficheiro ENV, que tem a informação de quais os elementos gráficos que vão ser visualizados no sinóptico;
- O ficheiro RUL, que tem o conjunto de regras que descreve o comportamento dos elementos gráficos com base nas características do modelo comportamental;
- O ficheiro PNML, que tem a representação da rede de Petri, e que é necessário aqui para obter informação sobre a capacidade dos lugares e sobre as entradas e saídas;
- O ficheiro VHDL, que tem o código de execução do controlador, não integrando obviamente a capacidade de actualização de sinóptico.

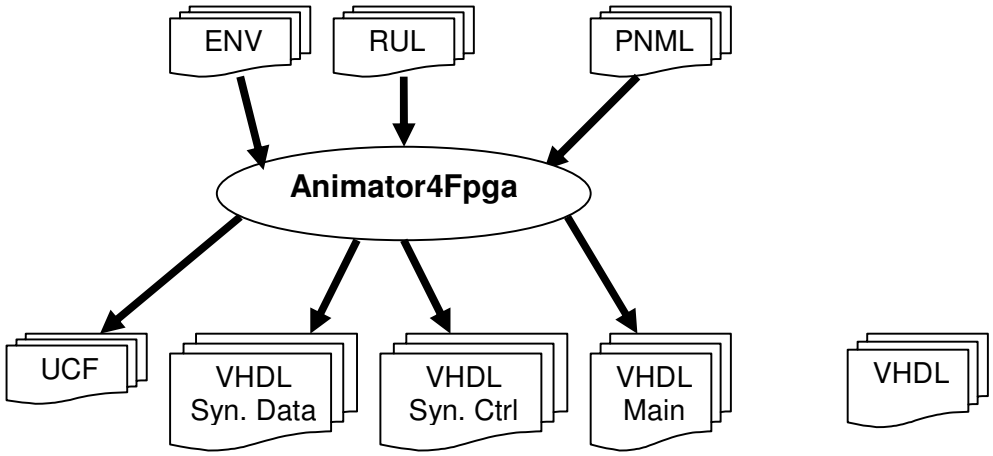


Figura 3.6. Arquitectura geral do *Animator4FPGA*

A aplicação Animator4Fpga será desenvolvida no âmbito desta dissertação e tem o objectivo:

- de gerar os ficheiros VHDL responsáveis pela actualização do sinóptico (*VHDL Syn. Data* e *VHDL Syn. Ctrl*),
- garantir a sua ligação ao ficheiro VHDL de execução da rede (a implementação da ligação está no ficheiro *VHDL Main*)
- e de produzir a definição de quais os pinos da FPGA que vão ser usados (ficheiro *UCF*).

Capitulo 4 – Caracterização da Plataforma de Execução

Um dos objectivos do trabalho é que o controlador gerado seja implementado em plataformas reconfiguráveis baseadas em FPGA integrando interfaces gráficas dedicadas (LCD) e interfaces de entrada/ saída.

Neste capítulo apresenta-se a arquitectura da plataforma de execução, a arquitectura da plataforma de gravação da EEPROM e o hardware utilizado nestas plataformas.

O capítulo inicia-se apresentando as características da plataforma de implementação seleccionada. A plataforma irá ter dois modos de funcionamento, correspondendo a duas configurações distintas. O primeiro modo de funcionamento permitirá a transferência de imagens para uma memória EEPROM e corresponde a uma configuração da plataforma descrita na secção 4.2. O segundo modo de funcionamento permitirá a utilização em "modo normal", isto é, considerando uma configuração suportando a execução do modelo de controlo e actualização do sinóptico associado no LCD.

4.1. *Hardware*

Hardware utilizado nas plataformas foi:

- Spartan-3 Starter Kit Board,
- EEPROM, AT28C64B,
- LCD, CMG128240-02.

A escolha por estes componentes deveu-se aos seguintes factores:

- Baixo custo,
- Fácil acesso,

- Tecnologia conhecida, o que permitiu não aumentar demasiado o tempo de desenvolvimento,
- Dimensões reduzidas, o que torna a solução compacta,
- Velocidade de processamento.

4.1.1. Spartan-3 Starter Kit Board

Dos componentes apresentados nas arquitecturas anteriores, estão integrados nesta plataforma [Spartan-3, 2004] os seguintes componentes: a FPGA (Xilinx XC3S200 Spartan-3), a memória SRAM (IS61LV25616AL-10T), os interfaces de entrada/saída e o interface RS232.

Da memória SRAM [RAM, 2006] destacam-se as seguintes características:

- Capacidade de 256k x 16,
- Memória paralela, com 16 bits de dados,
- Tempo de acesso aos dados 10, 12ns,

4.1.2. EEPROM

A EEPROM utilizada foi a AT28C64B da Atmel [EEPROM, 2006], é utilizada para armazenar as imagens (a imagem de fundo e as imagens animadas). Desta destacam-se as seguintes características:

- Capacidade de 8k bytes,
- Memória paralela, com 8 bits de dados,
- Tempo de leitura 150ns,
- Tempo de escrita 10ms,
- É acedida como uma memória RAM estática.

4.1.3. LCD

O LCD utilizado foi o CMG128240-02 com o controlador T6963C [LCD Controller, 1995], do qual se destacam as seguintes características:

- Interface paralelo de 8 bits simples,
- Diversos formatos de visualização, neste trabalho foi escolhido o modo gráfico com resolução 240x128,
- Dispõe de memória texto e memória gráfica.

4.2. Arquitectura da Plataforma de Gravação da EEPROM

Na figura 4.1. apresenta-se a arquitectura da plataforma de gravação da EEPROM, constituída por uma FPGA, memória EEPROM e interface RS232. O gravador é implementado na FPGA e recebe as imagens via RS232.

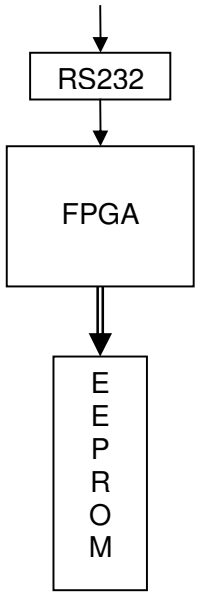


Figura 4.1. Arquitectura da plataforma de gravação da EEPROM

4.3. Arquitectura da Plataforma de Execução

Na figura 4.2. apresenta-se a arquitectura da plataforma de execução, constituída por uma FPGA, um LCD, memória EEPROM, memória SRAM, e interfaces de entrada/saída, com as seguintes funções:

- Na FPGA é implementado o controlador com o respectivo sinóptico,
- no LCD é apresentado o sinóptico,
- na memória EEPROM estão armazenadas as imagens,
- a memória SRAM é usada para construir as imagens (imagem de fundo mais imagens animadas) antes de serem apresentadas no LCD,
- os interfaces de entrada/saída permitem ao controlador implementado na FPGA comunicar com o mundo físico.

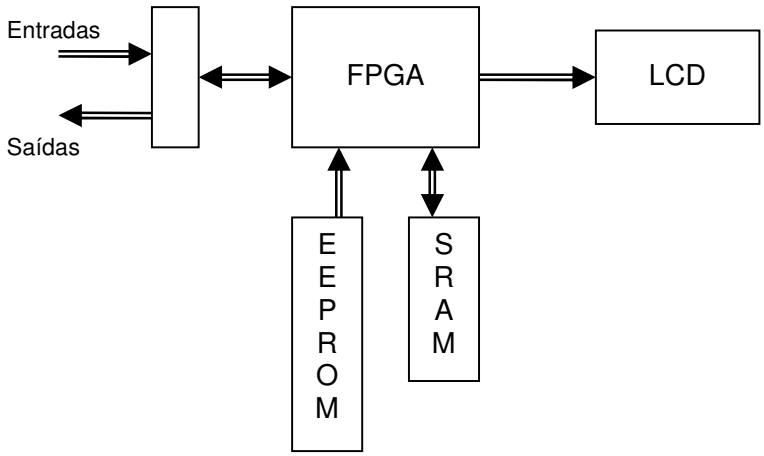


Figura 4.2. Arquitectura da plataforma de execução

4.4. Ambiente de desenvolvimento e de configuração da plataforma reconfigurável

O ambiente utilizado para configurar a plataforma via JTAG (*Joint Test Action Group*), foi o Project Navigator, versão 6.2i, da Xilinx, Inc.

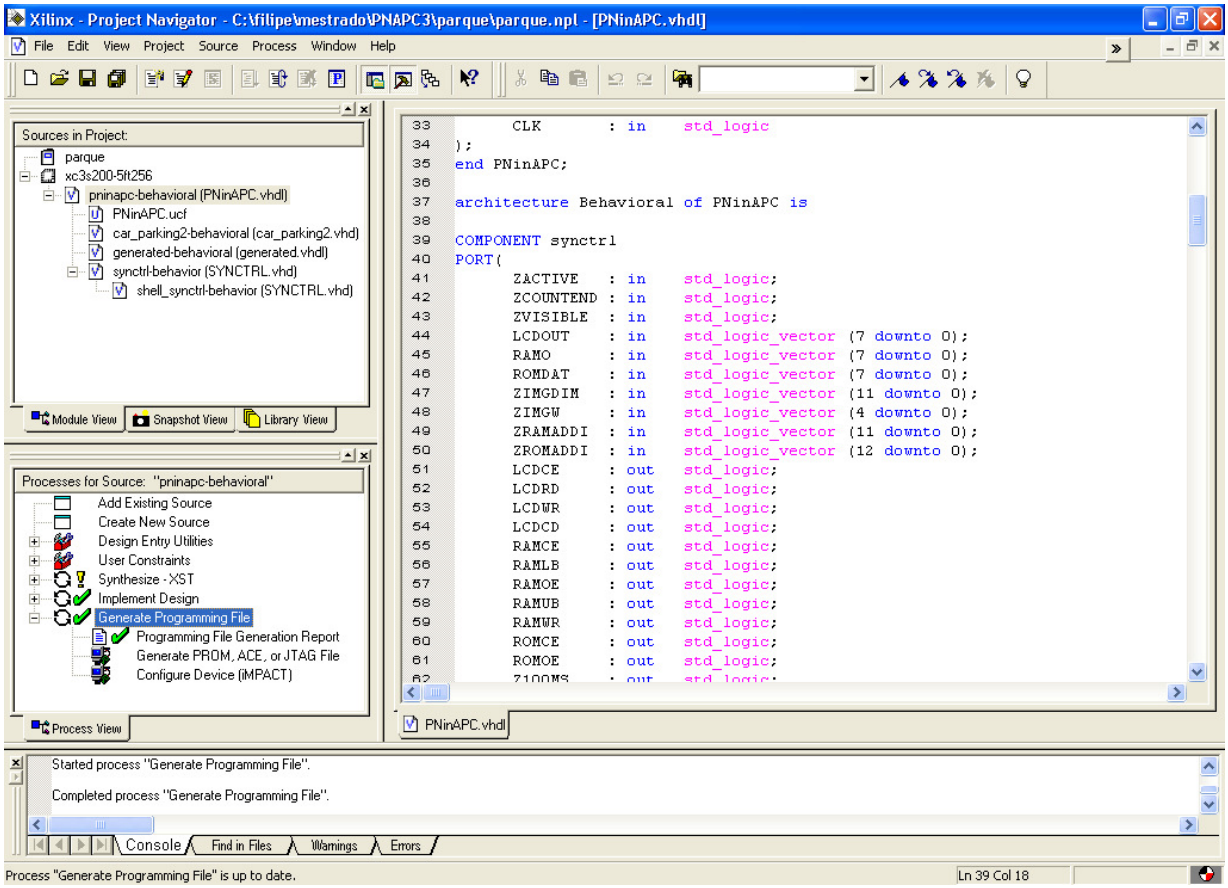


Figura 4.3. Project Navigator da Xilinx

Capitulo 5 – Implementações

Neste capitulo apresentam-se as implementações do controlador integrando interface gráfica, a implementação do gravador da EEPROM e a implementação do Animator4FPGA.

5.1. Implementação do Controlador Integrando Interface Gráfica

Na figura 5.1. apresenta-se o diagrama de blocos do controlador integrando interface gráfica. Este diagrama é composto pelo controlador do sistema e pelo sinóptico que tem a função de fazer a animação gráfica do sistema controlado.

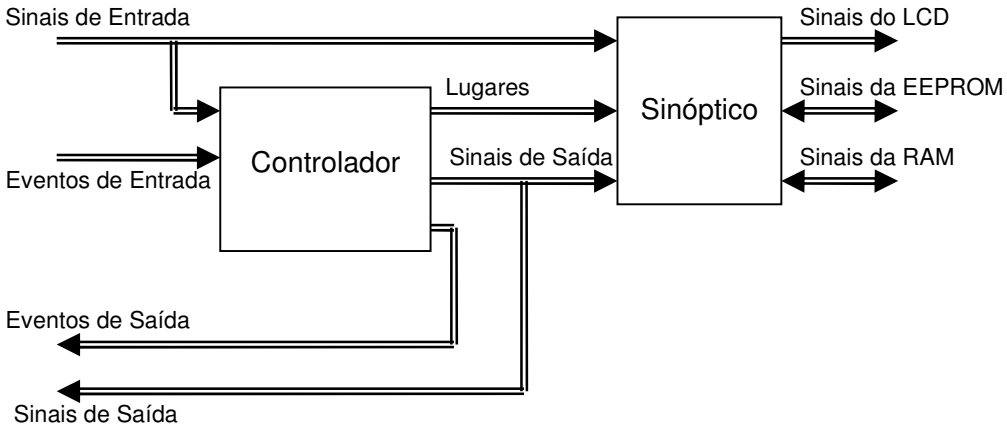


Figura 5.1. Diagrama de blocos do controlador integrando interface gráfica

A geração do código de implementação do controlador é feita pela ferramenta PNML2VHDL referida anteriormente, não sendo por isso descrita neste trabalho. De seguida descreve-se a implementação do sinóptico.

5.1.1. Implementação do Sinóptico

O sinóptico, como o nome indica tem a função de dar uma visão do que se está a passar no sistema e no respectivo controlador. Para tal, recebe como entradas os sinais de entrada/saída

e os lugares da rede de Petri que modelam o controlador do sistema, e apresenta no LCD imagens que dão a visão do que se está a passar. As imagens que vão ser mostradas no LCD estão armazenadas na EEPROM. A RAM é usada para construir as imagens antes de serem apresentadas no LCD.

O sinóptico proposto é constituído por duas partes, uma parte de dados e uma parte de controlo, como se pode ver na figura 5.2..

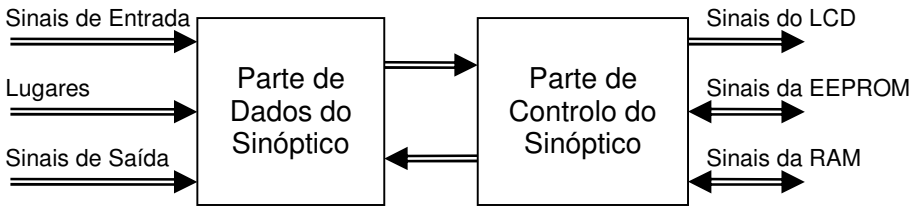


Figura 5.2. Diagrama de blocos do sinóptico

5.1.1.1. Implementação da Parte de Dados do Sinóptico

A parte de dados do sinóptico fornece à parte de controlo a seguinte informação:

- Localizações das imagens na EEPROM,
- Tamanho das imagens,
- Largura das imagens,
- Localizações das imagens no fundo das regras sem movimento,
- Localizações das imagens no fundo das regras com movimento,
- Visibilidades das imagens no fundo das regras sem movimento,
- Visibilidades das imagens no fundo das regras com movimento,
- Actividades das regras, isto é, se as regras estão activas (com base nos lugares e sinais de entrada dá indicação se uma determinada regra está activa num determinado momento).

As localizações e visibilidades das imagens no fundo das regras com movimento variam de acordo com o tempo que passou desde que a respectiva regra ficou activa.

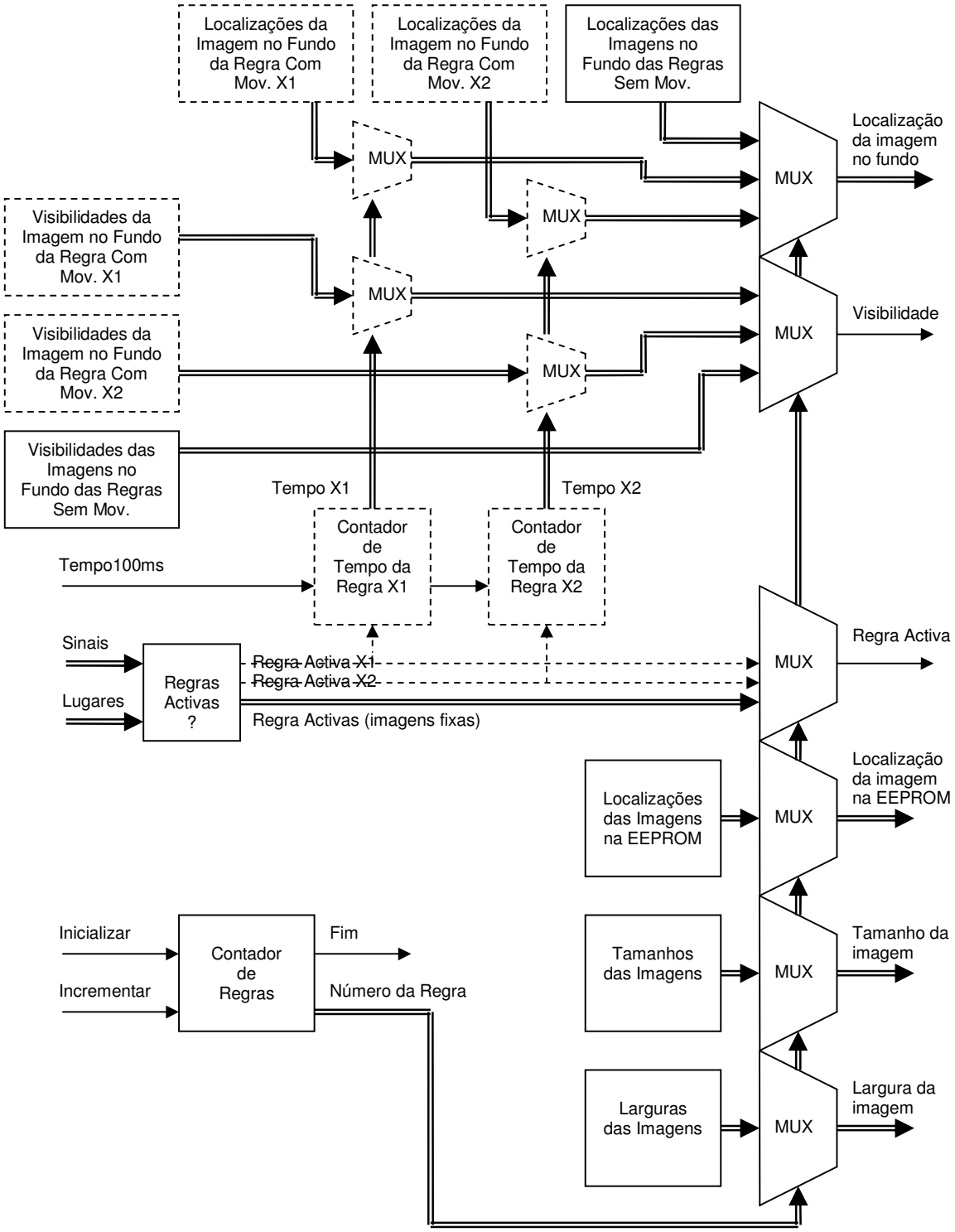


Figura 5.3. Diagrama de blocos da parte de dados do sinóptico

Observações relativas ao diagrama de blocos da figura 5.3.:

- o blocos “MUX” são multiplexeres,
- os blocos a tracejado são blocos relativos a regras com movimentos, só existem se existirem regras com movimentos. Neste exemplo considerou-se um exemplo em que existiam duas regras com movimento (a regra X1 e a regra X2),
- o contador de regras que é controlado pela parte de dados, tem a função de seleccionar nos multiplexeres:
    - qual a imagem que vai ser mostrada,
    - em que local do fundo a imagem vai ser mostrada,
    - se a imagem é visivel, se não estiver visivel não é mostrada,
    - se a regra está activa, se não estiver activa a imagem respectiva não é mostrada.,
- O bloco “Regras Activas?” tem a função de verificar os sinais de entrada/saída e os lugares da RdP de modo a indicar se cada regra está activa num determinado momento,
- O contador de tempo inicia a contagem quando uma regra fica activa.

5.1.1.2. Implementação da Parte de Controlo do Sinóptico

A parte de controlo do sinóptico tem as seguintes funções:

- Ler da parte de dados a informação descrita anteriormente,
- Ler da EEPROM as imagens,
- Construir na RAM a imagem que será mostrada no fundo do LCD,
- Mostrar no LCD a imagem construída.

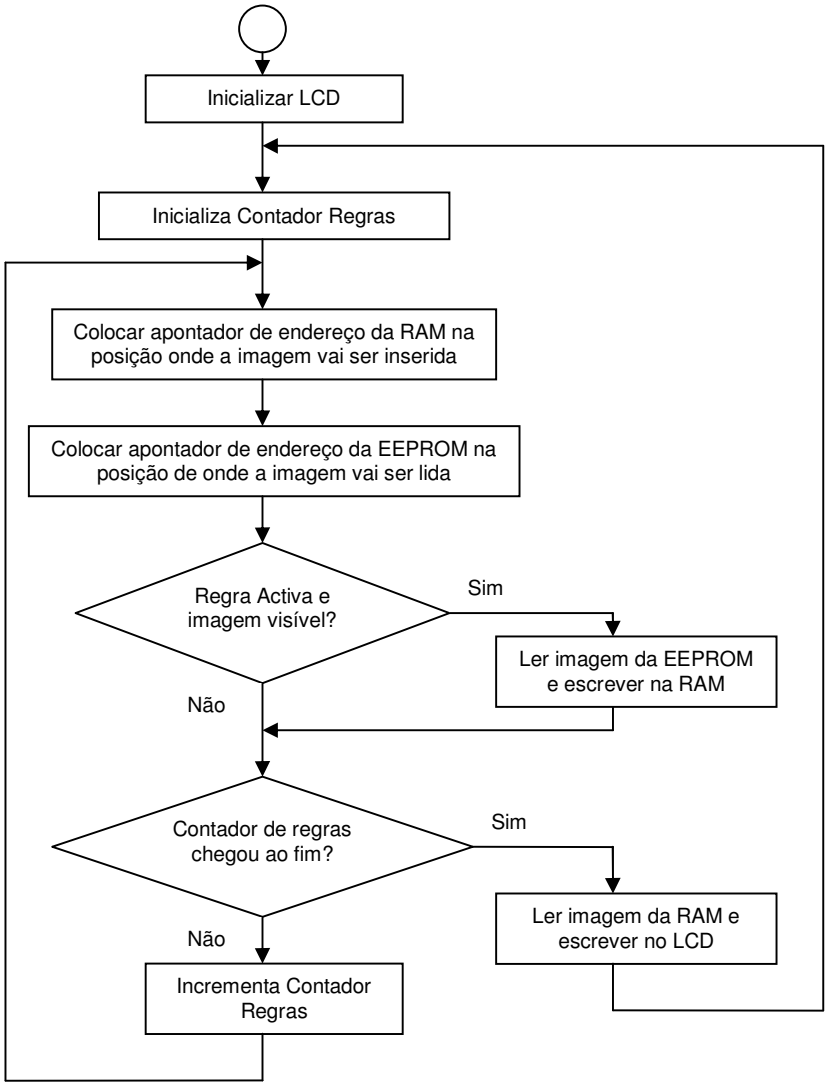


Figura 5.4. Fluxograma principal da parte de controlo do sinóptico

Na figura 5.4. pode ver-se de forma mais detalhada as acções realizadas pela parte de controlo e a ordem pela qual são realizadas:

1. inicializa o LCD,
2. inicializa o contador de regras existente na parte de dados do sinóptico,
3. coloca o apontador de endereço da RAM na posição onde a imagem vai ser inserida, que será a posição onde a imagem vai aparecer no LCD,
4. coloca o apontador de endereço da EEPROM na posição de onde a imagem vai ser lida,

5. verifica se a regra que está a ser avaliada está activa e se a imagem é visível, se sim lê a imagem da EEPROM e escreve-a na RAM (figura 5.5.),
6. verifica se o contador de regras chegou ao fim, isto é, se já verificou as regras todas,
7. se o contador de regras ainda não chegou ao fim, incrementa-o, e verifica a próxima regra, isto é, volta ao ponto 3.,
8. se o contador de regras chegou ao fim, copia a imagem construída na RAM para o LCD (figura 5.6.), e volta ao ponto 2..

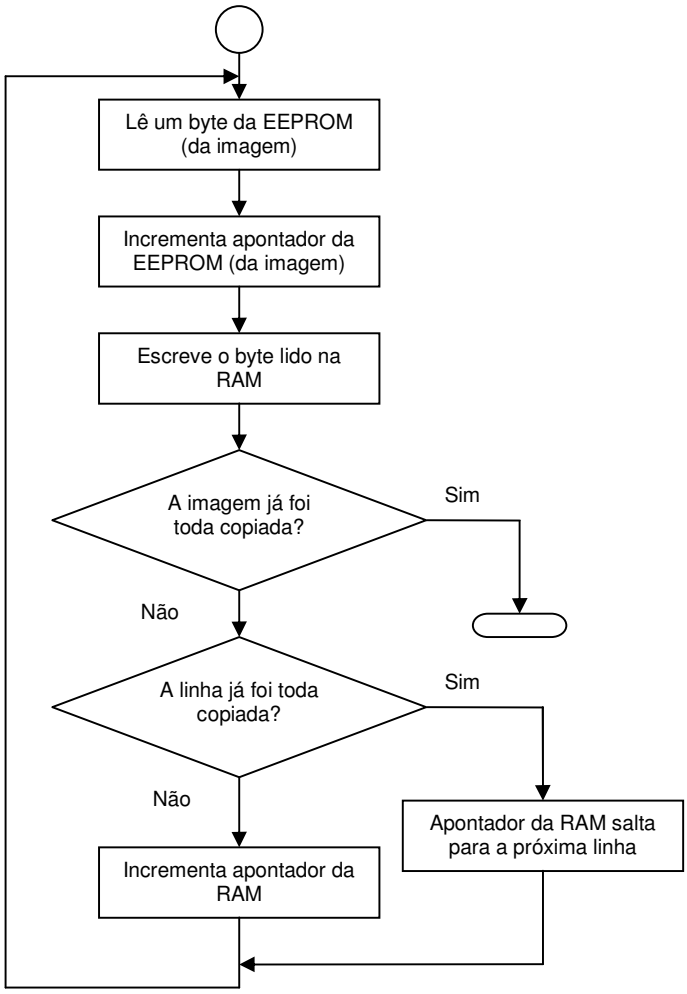


Figura 5.5. Fluxograma de leitura da EEPROM e escrita na RAM

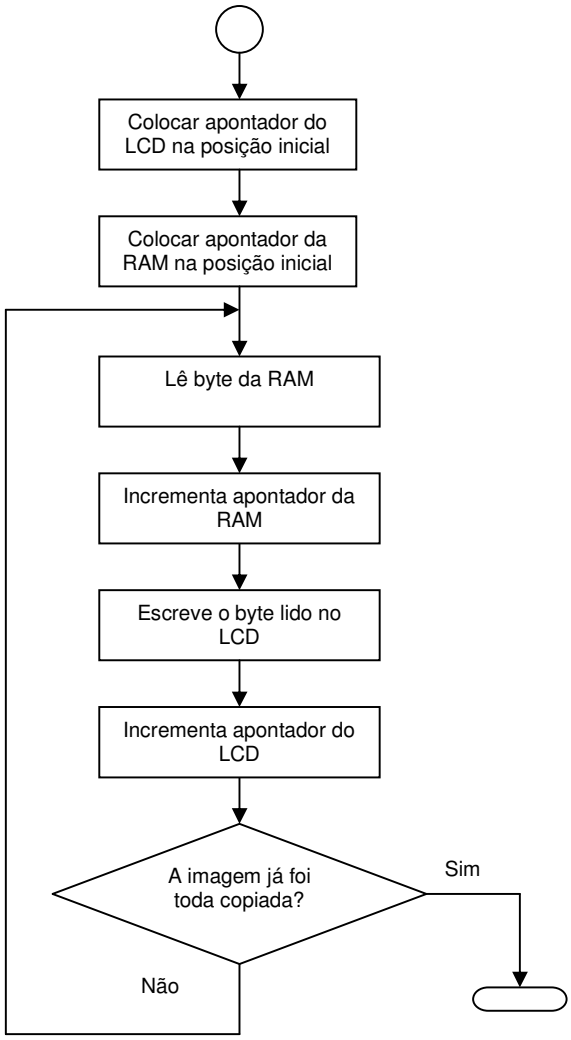


Figura 5.6. Fluxograma de leitura da RAM e escrita no LCD

Para além disto a parte de controlo tem ainda a função de gerar um sinal com um período de 100ms que é usado pela parte de dados para contabilizar o tempo.

A parte de controlo do sinóptico é sempre igual, independentemente da parte de dados. O que significa que o ficheiro que contém a descrição da implementação da parte de controlo é sempre o mesmo independentemente do controlador e da animação gráfica. Este ficheiro foi gerado automaticamente a partir do modelo em diagramas de estado pela ferramenta StateCAD da Xilinx.

Detalhes do LCD

Antes de começar a escrever na memória do LCD o que se pretende que seja mostrado, é necessária a sua inicialização, esta é feita apenas uma vez e após a sua ligação. Neste trabalho o LCD foi inicializado da seguinte forma:

- Definir o endereço inicial da área gráfica. Para se definir o endereço (0000H) é necessário:
    - *Status check* (verificar se o LCD está pronto para receber dados ou comandos),
    - *1st data* *Low addr byte* 00h (enviar o 1º byte de dados, que neste caso é o byte menos significativo do endereço, o 00h),
    - *Status check*,
    - *2st data* *High addr byte* 00h (enviar o 2º byte de dados, que neste caso é o byte mais significativo do endereço, o 00h)
    - *Status check*,
    - *Command* 42h (enviar o comando, que neste caso é o que define o endereço inicial da área gráfica, o 42h)
- Definir a área gráfica (128x240):
    - *Status check*,
    - *1st data* 1Eh (código que identifica a resolução da área gráfica),
    - *Status check*,
    - *2st data* 00h (sempre 00h)
    - *Status check*,
    - *Command* 43h
- Definir o modo (“OR” (texto ou gráfico), “CG-ROM” Mode (caso o modo texto fosse utilizado os caracteres utilizados eram os que estão na memória ROM interna)):
    - *Status check*,
    - *Command* 80h

- Definir o modo de visualização (gráfico ligado, texto desligado, sem cursor):
    - *Status check*,
    - *Command* 98h

Tendo sido feita a inicialização, antes de se começar a ler ou a escrever dados no LCD, é necessário garantir que o apontador deste está na posição certa, neste trabalho a posição é a 0000h (porque na inicialização se definiu este endereço como sendo o endereço inicial da área gráfica).

Para se definir qual o endereço onde se pretende começar a ler ou a escrever dados:

- *Status check*,
- *1st data* 00h ,
- *Status check*,
- *2st data* 00h,
- *Status check*,
- *Command* 24h.

Após isto é possível começar a escrever dados no LCD, para tal usa-se o seguinte comando, que para além de escrever o byte, ainda incrementa o endereço, para que o próximo byte seja escrito no endereço seguinte:

- *Status check*,
- *Data* XXh (XXh -> byte que se pretende enviar),
- *Status check*,
- *Command* C0h.

5.2. Implementação do Gravador de Imagens na EEPROM

É necessário gravar as imagens que virão a ser visualizadas na EEPROM, para tal desenvolveu-se um modulo com que se configura a FPGA. Recebe da linha série as imagens e grava-as na EEPROM.

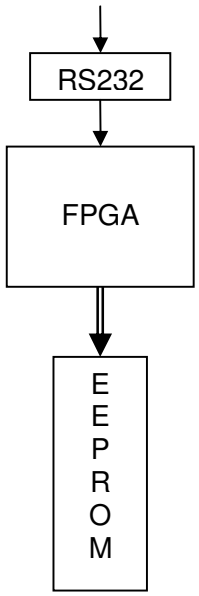


Figura 5.7. Diagrama do blocos gravador de imagens na EEPROM

O módulo desenvolvido para a FPGA tem o algoritmo apresentado na figura 5.8.. Foi gerado automaticamente a partir do modelo em diagramas de estado pela ferramenta StateCAD da Xilinx.

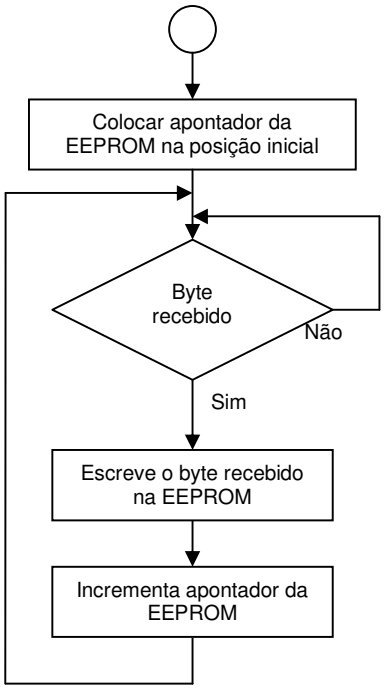


Figura 5.8. Algoritmo do módulo de gravação das imagens na EEPROM

Para receber um byte o que faz é gerar um clock 4 vezes superior ao ritmo de recepção, e escutar RX, quando RX fica a zero durante dois clocks significa que chegou um start bit, depois espera 4 clocks e lê o 1º bit, espera mais 4 bits e lê o 2º bit, e faz isto até ao stop bit.

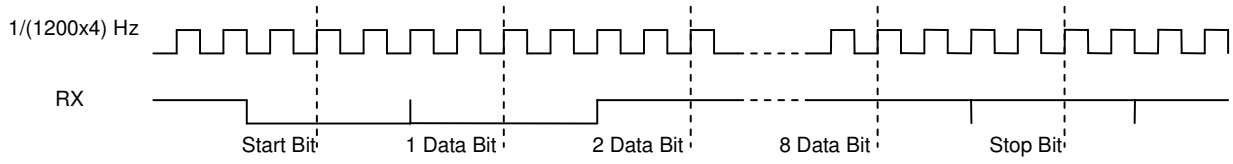


Figura 5.9. Escuta do sinal RX do interface de comunicação série RS232

A ferramenta que se usou para enviar o ficheiro que contém as imagens pela linha série do PC para a FPGA foi o “Hercules SETUP utility by HW-group.com”.

5.3. Implementação do Animator4FPGA

O Animator4FPGA foi desenvolvido utilizando o Microsoft Visual Studio 2005, que usa o Microsoft .NET Framework 2.0, também usado noutras ferramentas do FORDESIGN. A linguagem de desenvolvimento foi o C#.

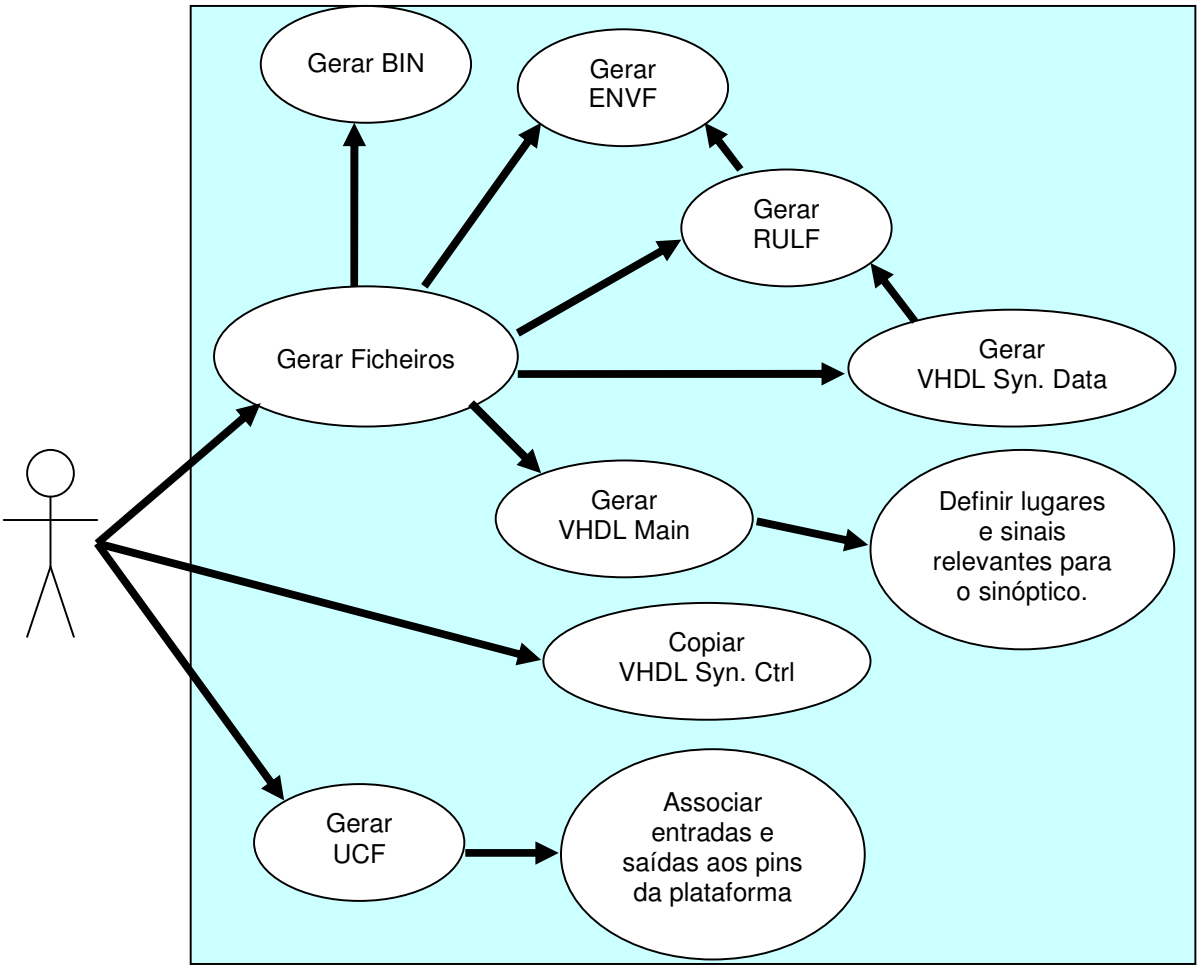


Figura 5.10. Diagrama de casos de uso do *Animator4FPGA*

Tal como descrito no capítulo 3, a função do Animator4FPGA é gerar os ficheiros VHDL e o ficheiro UCF para configurar a FPGA, bem como o ficheiro BIN com as imagens para gravar na EEPROM.

Na figura 5.10. pode ver-se o diagrama de casos de uso, onde:

- O caso de uso “Gerar Ficheiros” tem como pré condições a selecção dos ficheiros *enviroment* e *rules* criados pela aplicação Animator, e o ficheiro PNML criado pela aplicação Snoopy-IOPT. A primeira acção a realizar neste caso de uso é a definição os lugares e os sinais relevantes para o sinóptico, de seguida e com base nos ficheiros seleccionados e nas condições definidas podem ser gerados os ficheiros BIN, ENVF, RULF, VHDL Syn. Data e VHDL Main.
- O caso de uso “Copiar VHDL Syn. Ctrl” é uma simples acção de copiar o ficheiro “VHDL Syn. Ctrl” para a directoria de destino. Este ficheiro é utilizado em todas as aplicações geradas, pelo que bastará ser copiado.
- O caso de uso “Gerar UCF”, tem como pré condição que o caso de uso “Gerar Ficheiros” já tenha sido executado. Este caso de uso necessita que seja feita a associação entre as entradas e as saídas e os pins da plataforma, com esta informação é gerado o ficheiro UCF.

O Animator4FPGA gera 6 ficheiros:

- VHDL Syn. Data
- VHDL Main
- UCF
- BIN
- ENVF
- RULF

O ficheiro VHDL Syn. Data contém a parte de dados do sinóptico. Estes dados são as posições de memória onde estão guardadas as imagens, as características das imagens e as regras que controlam o posicionamento e a deslocação das imagens.

O ficheiro VHDL Main faz a ligação dos ficheiros VHDL Syn. com o ficheiro VHDL que implementa o controlador.

O ficheiro UCF é um ficheiro que contém a informação dos pins e é utilizado para a implementação na plataforma especifica.

O ficheiro BIN contém as imagens, será usado para as gravar na EEPROM.

Os ficheiros ENVF e RULF, são auxiliares. O primeiro tem por base o ficheiro ENV e detêm as características e a localização na memória de imagens das imagens que vão ser utilizadas pelo sinóptico ver tabela 5.1.. O segundo tem por base o ficheiro RUL, e apresenta as regras transformadas que serviram de base à criação do ficheiro VHDL Syn. Data, ver tabela 5.2..

***Tag* XML**	**Descrição**
<Backgrounds>	Elemento que identifica os ambientes
<Background>	Elemento que identifica o início de um ambiente
<Name>	Nome do ambiente
<SourceStart>	Posição onde a imagem de fundo do ambiente está guardada
<SourceWidth>	Largura da imagem de fundo do ambiente
<SourceLength>	Tamanho da imagem de fundo do ambiente
<MonitorXLocation>	Coordenada X da localização do ambiente
<MonitorYLocation>	Coordenada Y da localização do ambiente
<Animated>	Elemento que identifica o início de uma imagem animada
<Anim_Name>	Elemento que identifica o nome de uma imagem animada
<Anim_SourceStart>	Posição onde a imagem animada está armazenada
<Anim_SourceWidth>	Largura da imagem animada
<Anim_SourceLength>	Tamanho da imagem animada
<Anim_BGLocation>	Localização da imagem animada no fundo
<Anim_Visible>	Visibilidade da imagem animada
<Anim_Transparency>	Transparência da imagem animada
<Anim_Zoom>	Zoom da imagem animada

Tabela 5.1. Detalhes do ficheiro ENVF

Tag XML	Descrição
<Rules>	Elemento que identifica as regras
<Rule>	Elemento que identifica o início de uma regra
<RuleID>	Identificação da Regra
<if>	Característica estática que se avalia da RdP
<thenImgAddrStart>	Posição onde a imagem animada está armazenada
<thenImgLength>	Largura da imagem animada
<thenImgWidth>	Tamanho da imagem animada
<thenTransparency>	Transparência da imagem animada
<thenLocation>	Localização da imagem animada no fundo
<thenZoom>	Zoom da imagem animada
<thenVisible>	Visibilidade da imagem animada
<thenTime>	Tempo que a imagem animada demora a chegar ao "thenLocation"

Tabela 5.2. Detalhes do ficheiro RULF

Para além dos ficheiros que o Animator4FPGA gera, existe ainda outro ficheiro necessário para o funcionamento do controlador integrando animação gráfica: o ficheiro VHDL Syn. Ctrl, que é responsável pela leitura e escrita das imagens em memória, pela escrita no dispositivo gráfico, e pelo controlo do ficheiro VHDL Syn. Data. Este ficheiro é sempre igual, independentemente do sinóptico, e por isso basta ser copiado (não necessita de ser gerado).

De modo a facilitar alterações futuras na aplicação e visto que esta lê ficheiros (ENV, RUL, PNML) gerados por outras aplicações, os métodos responsáveis pela leitura destes ficheiros foram implementados numa DLL independente do resto do código, de modo que, se houver alterações num destes ficheiro, deverá será apenas necessário alterar esta DLL.

Capitulo 6 – Exemplo de Aplicação

6.1. Apresentação do Problema

O objectivo é desenvolver um controlador de um parque de estacionamento que integre funções de actualização do sinóptico, implementado numa plataforma reconfigurável baseada numa FPGA.

O parque de estacionamento tem as seguintes características:

- Capacidade igual a 3 lugares;
- Uma entrada;
- Uma saída;
- Um sensor de presença de carro e um de retirada de bilhete de acesso, na entrada;
- Um sensor de presença de carro e um de pagamento efectuado, na saída.

Nas secções seguintes apresenta-se, de forma breve, a sequência de procedimentos que permitem realizar a implementação do controlador referido na plataforma descrita.

6.2. Modelação através de uma RdP

Utilizado o Snoopy IOPT, foi primeiro editada a rede de Petri e depois gerado o ficheiro PNML com a sua representação.

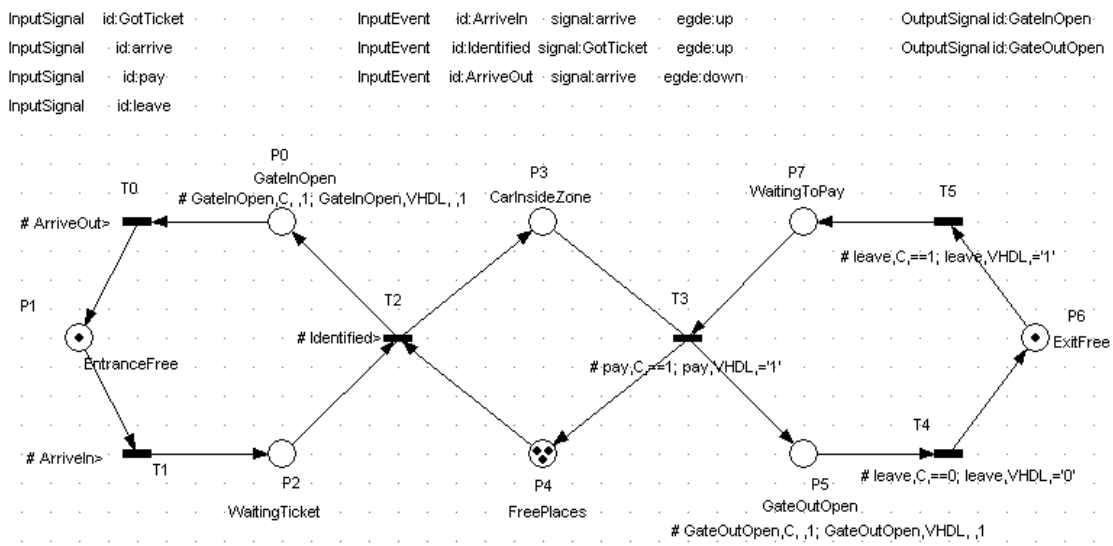


Figura 6.1. Exemplo de uma rede de Petri editada no Snoopy IOPT

6.3. Geração do VHDL a partir do PNML

Utilizando a ferramenta PNML2VHDL, foi gerado o código VHDL que implementa o controlador do parque.

6.4. Animação da RdP do Controlador

Para animar a rede de Petri utilizam-se em sequência duas aplicações: a Animator e a Animator4Fpga desenvolvida nesta dissertação.

O primeiro conjunto de procedimentos realiza-se através da aplicação Animator, começando-se por adicionar uma imagem ao fundo (background) e adicionar as imagens animadas (imagens que vão aparecer, desaparecer e ter movimento), como apresentado nas Figuras 6.2 e 6.3..

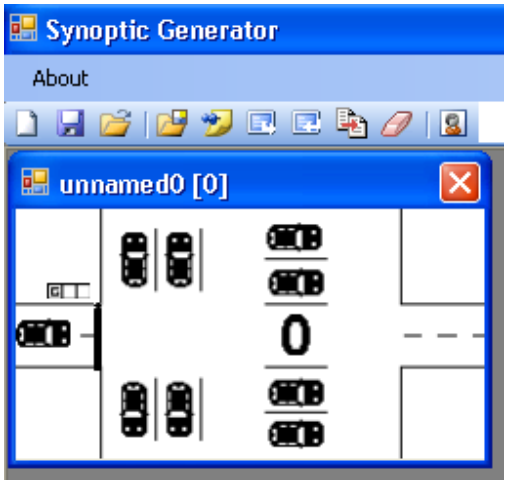


Figura 6.2. Exemplo de um *Background* no *Animator*

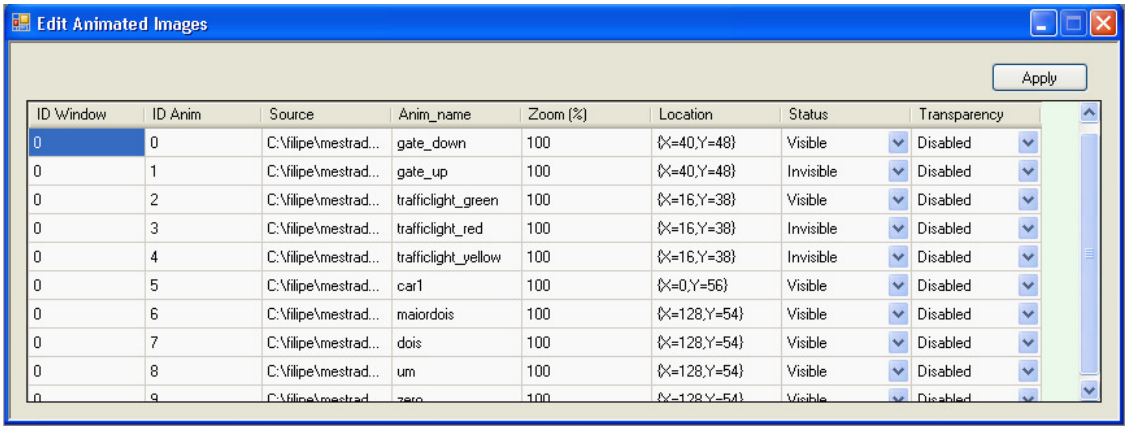
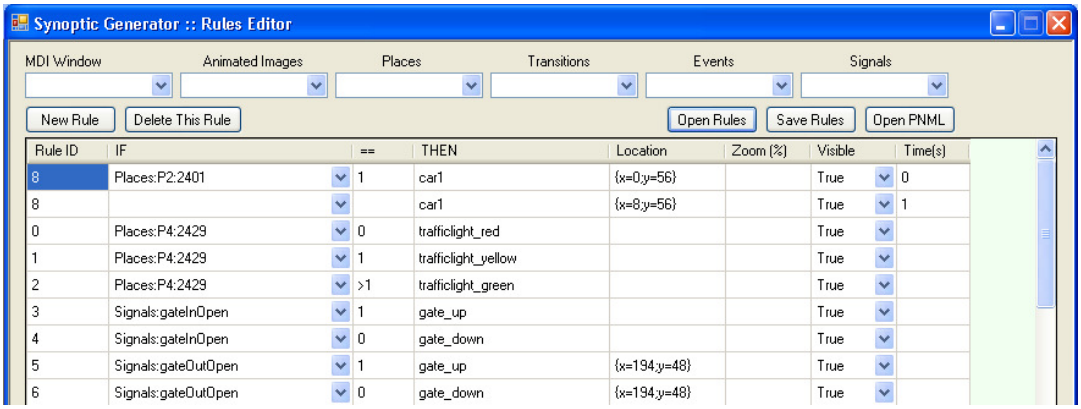


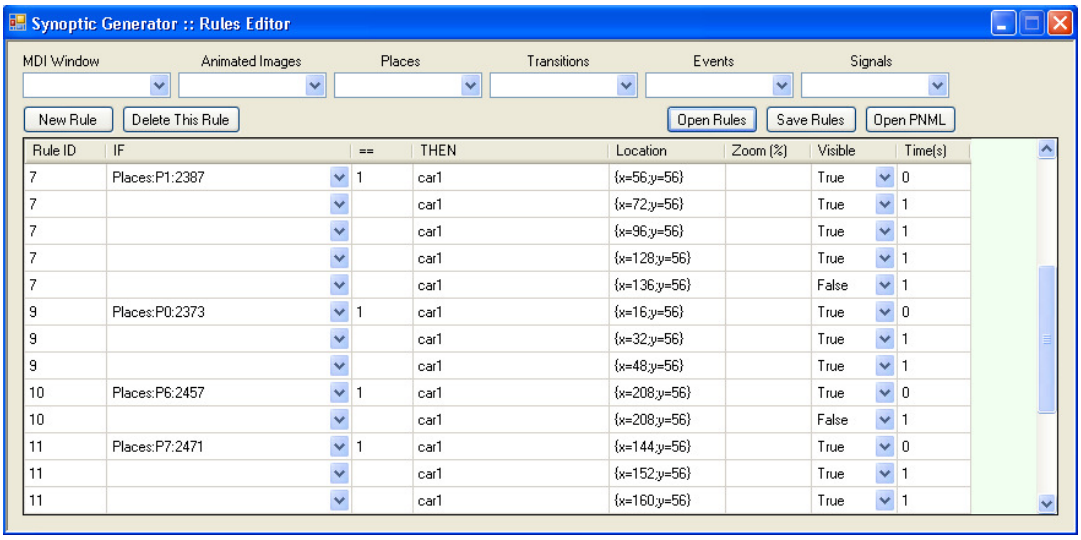
Figura 6.3. Exemplo de uma lista de imagens animadas no *Animator*

O passo seguinte é a definição de regras que vão definir o comportamento das imagens animadas. As Figuras 6.4, 6.5 e 6.6 ilustram a sua criação através da utilização de interfaces de diálogo dedicados.



Rule ID	IF	==	THEN	Location	Zoom (%)	Visible	Time(s)
8	Places:P2:2401	1	car1	{x=0;y=56}		True	0
8			car1	{x=8;y=56}		True	1
0	Places:P4:2429	0	trafficlight_red			True	
1	Places:P4:2429	1	trafficlight_yellow			True	
2	Places:P4:2429	>1	trafficlight_green			True	
3	Signals:gateInOpen	1	gate_up			True	
4	Signals:gateInOpen	0	gate_down			True	
5	Signals:gateOutOpen	1	gate_up	{x=194;y=48}		True	
6	Signals:gateOutOpen	0	gate_down	{x=194;y=48}		True	

Figura 6.4. Exemplo de uma lista de regras animadas no *Animator*



Rule ID	IF	==	THEN	Location	Zoom (%)	Visible	Time(s)
7	Places:P1:2387	1	car1	{x=56;y=56}		True	0
7			car1	{x=72;y=56}		True	1
7			car1	{x=96;y=56}		True	1
7			car1	{x=128;y=56}		True	1
7			car1	{x=136;y=56}		False	1
9	Places:P0:2373	1	car1	{x=16;y=56}		True	0
9			car1	{x=32;y=56}		True	1
9			car1	{x=48;y=56}		True	1
10	Places:P6:2457	1	car1	{x=208;y=56}		True	0
10			car1	{x=208;y=56}		False	1
11	Places:P7:2471	1	car1	{x=144;y=56}		True	0
11			car1	{x=152;y=56}		True	1
11			car1	{x=160;y=56}		True	1

Figura 6.5. Exemplo de uma lista de regras animadas no *Animator* (Continuação)

12	Places:P5:2443	1	car1	{x=168;y=56}		True	0
12			car1	{x=184;y=56}		True	1
12			car1	{x=200;y=56}		True	1
13	Places:P4:2429	0	zero			True	
14	Places:P4:2429	1	um			True	
15	Places:P4:2429	2	dois			True	
16	Places:P4:2429	>2	maiordois			True	

Figura 6.6. Exemplo de uma lista de regras animadas no *Animator* (Continuação)

O segundo conjunto de procedimentos utiliza a aplicação Animator4FPGA, gerando-se os seguintes ficheiros:

- Controller.bin (para configurar a EEPROM)

- ENVF (ficheiro contendo as características do ambiente gráfico; ficheiro gerado disponível no Anexo B)
- RULF (ficheiro contendo as regras de associação das características do modelo com as características para animação gráfica; ficheiro gerado disponível no Anexo C)
- Syndata.vhdl (parte de dados do sinóptico)
- PNinAPC.vhdl (ficheiro VHDL de topo de hierarquia na descrição produzida)
- PNinAPC.ucf (atribuição de pinos)

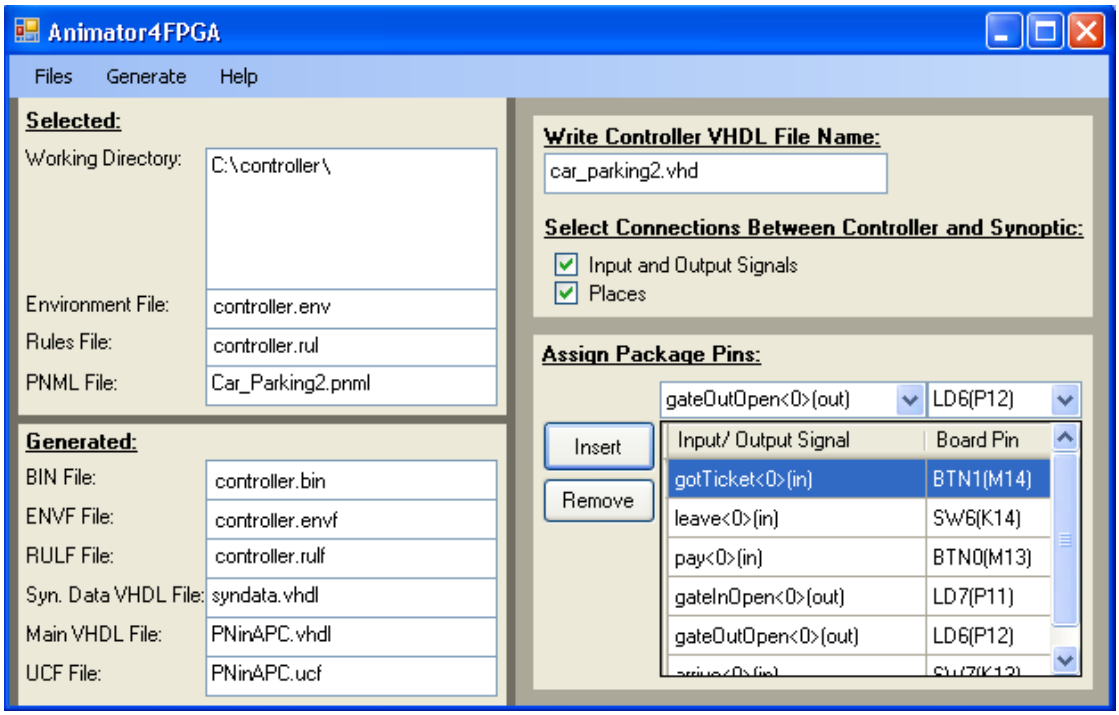


Figura 6.7. Exemplo do *Animator4FPGA*

Juntando aos ficheiros VHDL gerados os seguintes ficheiros VHDL:

- Car_parking.vhd (controlador gerado pelo PNML2VHDL)
- Synctrl.vhdl (parte de controlo do sinóptico, sempre igual para todos os casos de aplicação)

obtêm-se os ficheiros necessários para implementar o controlador integrando animação gráfica. Estes são adicionados a um projecto do *Project Navigator* da *Xilinx*, que gera o ficheiro para configurar a FPGA da plataforma reconfigurável. A EEPROM, cujo conteúdo armazena as imagens, é gravada por uma configuração do sistema desenvolvido para o efeito, e descrita no capítulo 5.

Na figura 6.8. apresenta-se uma fotografia da plataforma reconfigurável desenvolvida, e onde foi implementado o exemplo aqui descrito.

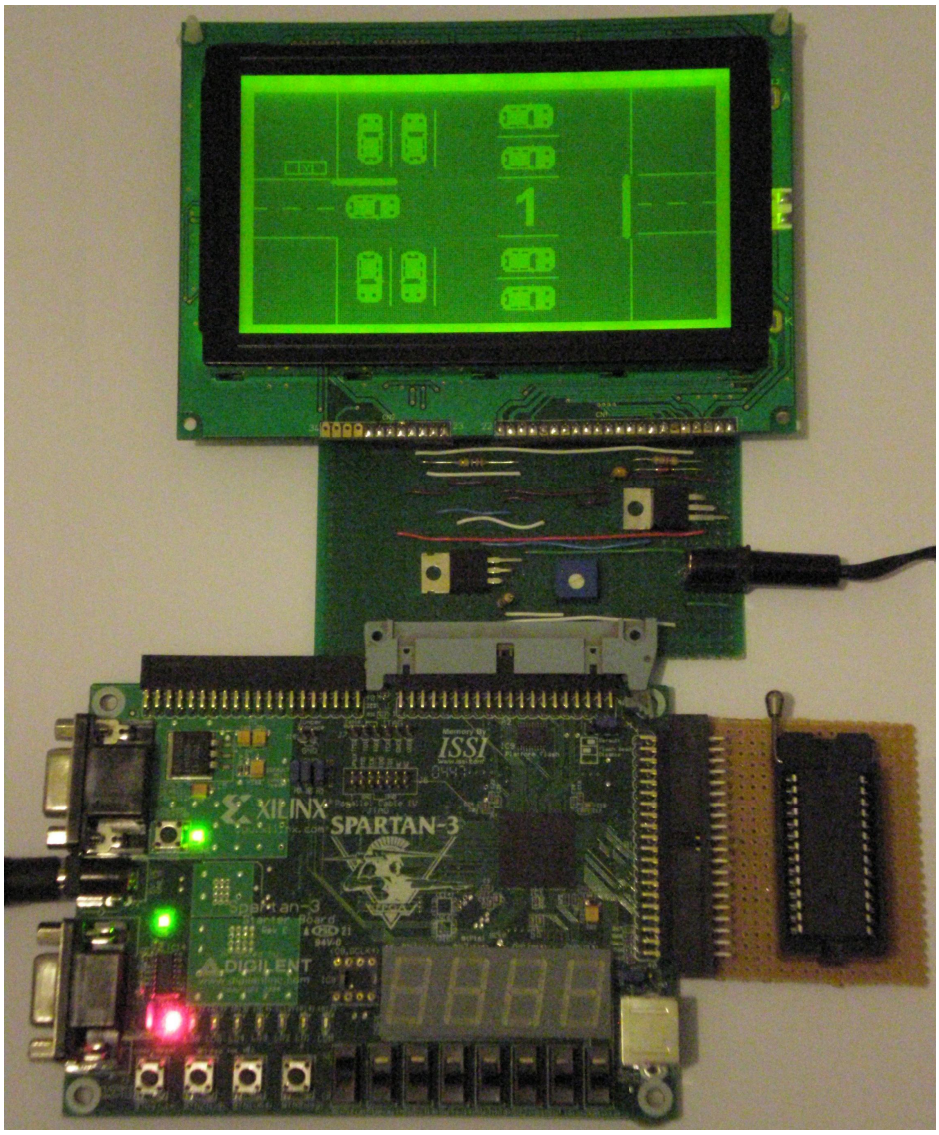


Figura 6.8. Fotografia da plataforma reconfigurável desenvolvida

Capitulo 7 – Conclusões e Perspectivas Futuras

A primeira conclusão do trabalho é o facto de não existirem (que se tenha conhecimento) ferramentas de geração automática que cumpram os objectivos propostos nesta dissertação, permitindo aliar atitudes de desenvolvimento baseadas em modelos com ferramentas de geração automática de código para o desenvolvimento deste tipo de sistemas de controlo embutido integrando interface para visualização de sinóptico associado.

As redes de Petri IOPT mostraram-se adequadas à modelação de sistemas, possibilitando a sua implementação através de ferramentas de geração automática.

Outra conclusão relevante está relacionada com a redução do tempo de desenvolvimento de controladores que integrem animação gráfica em plataformas reconfiguráveis baseadas em FPGA. Na realidade, tendo disponível o ambiente de desenvolvimento descrito, a criação de um sistema de controlo e visualização do estado de um determinado processo pode ser realizada muito rapidamente, uma vez que todas as tarefas de desenvolvimento de código foram substituídas pela utilização de interfaces específicos para a sua caracterização.

Esta metodologia de desenvolvimento de controladores elimina erros decorrentes da geração manual de código, reduzindo assim os erros de desenvolvimento à etapa de modelação. Estes últimos existem independentemente do método de gerar o código.

Futuramente, considera-se de elevado interesse permitir que os controladores desenvolvidos pudessem comunicar remotamente com outros controladores, possibilitando o desenvolvimento de sistemas de controlo distribuídos, como é comum nos sistemas SCADA.

A ferramenta Animator4FPGA, o exemplo apresentado, bem como outros exemplos, podem ser encontrados na página http://www.uninova.pt/fordesign/Animator4FPGA.htm

Bibliografia

[Barros & Gomes, 2004]
João Paulo Barros, Luís Gomes. "Operational PNML: Towards a PNML Support for Model Construction and Modification", proceedings of the Workshop on Definition, Implementation and Application of a Standard Interchange Format for Petri Nets, satelite workshop of The 25th International Conference on Application and Theory of Petri Nets (ICATPN 2004), Bologna, Italy, June 26, 2004.

[BRITNeY Suite, 2008]
BRITNeY Suite website http://wiki.daimi.au.dk/britney/britney.wiki . Março, 2008.

[CPN Tools, 2008]
CPNTools website www.daimi.au.dk/CPNTools . Março, 2008.

[Fernandes et al., 1997]
J.M. Fernandes, M. Adamski, A.J. Proenca: VHDL generation from hierarchical Petri net specifications of parallel controllers. IEE Proceedings: Computers and Digital Techniques 144, 2 (1997) 127-137

[FORDESIGN, 2007]
FORDESIGN – Formal Methods for Embedded Systems Co-Design. 2007. R&D project POSC/EIA/61364/2004 sponsored by FCT - Fundação para a Ciência e a Tecnologia. www.uninova.pt/fordesign . Dezembro, 2007.

[Gomes, Costa & Meira, 2005]
Luis Gomes, Anikó Costa, and Paulo Meira, “From Use Cases to building monitoring systems through Petri nets,” in 10thISIE’2005 - 2005 IEEE International Symposium on Industrial Electronics, Dubrovnik, Croatia, June 2005.

[Gomes et al., 2005]
Luís Gomes, João Paulo Barros, Anikó Costa, Rui Pais, Filipe Moutinho. 2005. “Formal methods for Embedded Systems Co-design: the FORDESIGN project”. ETFA 2005. 10th IEEE Conference on Volume 1, Issue, 19-22 Sept. 2005. Catania, Italy.

[Gomes et al., 2007a]
Luís Gomes, Anikó Costa, João Paulo Barros, Rui Pais, Tiago Rodrigues, Richard Ferreira. “Petri Net based Building Automation and Monitoring System”; INDIN'2007 - 5th IEEE International Conference on Industrial Informatics, 23-26 July 2007, Vienna, Austria

[Gomes et al., 2007b]
Luís Gomes, João Paulo Barros, Anikó Costa, Ricardo Nunes ; “The Input-Output Place-Transition Petri Net Class and Associated Tools”; INDIN'2007 - 5th IEEE International Conference on Industrial Informatics, 23-26 July 2007, Vienna, Austria

[Gomes et al., 2007c]
Luís Gomes, Anikó Costa, João Paulo Barros, Paulo Lima; “From Petri net models to VHDL implementation of digital controllers”; IECON’2007 - The 33rd Annual Conference of the IEEE Industrial Electronics Society, November 5-8, 2007, The Grand Hotel, Taipei, Taiwan

[Gomes & Lourenço, 2008]
Luís Gomes, João Lourenço; “Petri net-based automatic generation GUI tools for embedded systems”; HSI’2008 – Conference on Human System Interaction; Krakow, Poland, May 25-27, 200;.

[Lourenço, 2008]
João Pedro Gouveia Lourenço; “Modelos comportamentais e redes de Petri na geração automática de animação de sinópticos”. Tese de Mestrado. 2008, Lisboa.

[Lourenço & Gomes, 2008]
João Lourenço, Luís Gomes. “Animated Synoptic Generator Framework for Input-Output Place-Transition Petri Net Models”; ICATPN’2008 - 29th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency. Xi’an, China, June 23-27, 2008; Lecture Notes in Computer Science, Springer.

[Marranghello et al., 2000]
N. Marranghello, J. Mirkowski, K. Bilinski: Synthesis of synchronous digital systems specified by Petri nets. In: Hardware Design and Petri Nets. Kluwer Academic Publishers (2000)

[Nios II Kit, 2008]
Nios II Embedded Evaluation Kit, Cyclone III Edition website
http://www.altera.com/products/devkits/altera/kit-cyc3-embedded.html?WT.mc_id=n8_en_cm_dl_bx_f_444.

[Nunes, Gomes & Barros, 2007]
Ricardo Nunes, Luís Gomes, João Paulo Barros. “A Graphical Editor for the Input-Output Place-Transition Petri Net Class”; ETFA'2007 - 12th IEEE Conference on Emerging Technologies and Factory Automation, September 25-28, 2007; Patras, Greece

[Pais, Barros & Gomes, 2005]
Rui Pais, João Barros, and Luis Gomes, “A Tool for Tailored Code Generation from Petri Net Models,” in 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2005)., Sep. 2005.

[Petri Net Kernel, 2002]
Petri Net Kernel website http://www2.informatik.hu-berlin.de/top/pnk/index.html Dezembro, 2007.

[PNML, 2004]
PNML, Petri Net Markup Language.2004. http://www.informatik.huberlin.de/top/pnml/about.html, Março, 2008.

[Reisig, 1985]
Wolfgang Reisig. Petri nets: an Introduction. 1985. ISBN 0-387-13723-8. Springer-Verlag New York, Inc.

[SNOOPY, 2007]
Data Structures and Software Dependability Brandenburg University of Technology Cottbus. S N O O P Y ’ s homepage. http://www-dssz.informatik.tu-cottbus.de/software/snoopy.html, Março, 2008.

[SNOOPY IOPT, 2008]
http://www.uninova.pt/fordesign/SnoopyIOPT.htm

[Soto & Pereira, 2001]
E. Soto, M, Pereira: Implementing a Petri net specification in a FPGA using VHDL. In Adamski, M., Wegrzyn, M., eds.: Proceedings of the International Workshop on Discrete-Event System Design - DESDes01. (2001)

[VDHL, 1988]
IEEE Standard VHDL Language Reference Manual, 1988, USA.

[Westergaard & Lassen, 2006]
Michael Westergaard, Kristian Lassen. The BRITNeY Suite Animation Tool. Petri Nets and Other Models of Concurrency 2006, volume 4024 of LNCS. Springer, 2006.

[Westergaard, 2006]
Michael Westergaard. 2006. The BRITNeY Suite: A Platform for Experiments. Aarhus University, Denmark.

Referências Bibliográficas Complementares

[EEPROM, 2006]
Data Sheet, AT28C64B, 64K (8K x 8) Parallel EEPROM with Page Write and Software Data Protection, ATMEL, 2006.

[LCD Controller, 1995]
Application Notes for the T6963C LCD Graphics Controller Chip, 1995.

[RAM, 2006]
Data Sheet, IS61LV25616AL, 256K x 16 High Speed Asynchronous Cmos Static RAM with 3.3V Supply, ISSI, 2006.

[Spartan-3, 2004]
Spartan-3 Starter Kit Board User Guide, Xilinx, 2004.

Anexos

Anexo A – Manual do Utilizador do Animator4FPGA

Selecção do Ambiente

A primeira acção a ser realizada pelo *Animator4FPGA* é seleccionar o ficheiro *environment* (*ENV*) gerado pelo *Animator*.

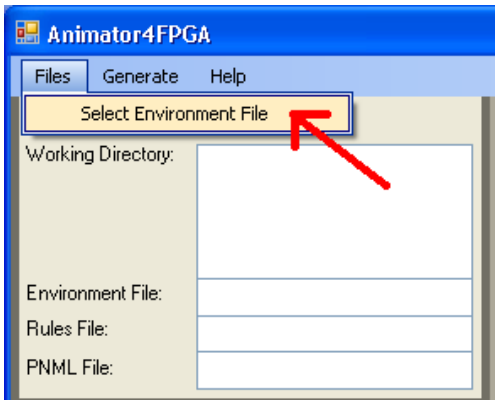


Figura 7.1. Selecção do ficheiro *ENV*

Na directoria onde está o ficheiro ENV deverá estar também um ficheiro RUL e um ficheiro PNML. São estes os ficheiros que o Animator4FPGA necessita.

Só pode existir um ficheiro de cada tipo na directoria seleccionada, caso contrário aparecerá uma mensagem de aviso.

Geração do ficheiro BIN e dos ficheiros VHDL

Para gerar os ficheiros, é necessário:

- Escrever o nome do ficheiro VHDL do controlador (gerado pelo PNML2VHDL),
- Seleccionar as ligações entre o controlador e o sinóptico,
- E por fim gerar os ficheiros BIN e VHDL.

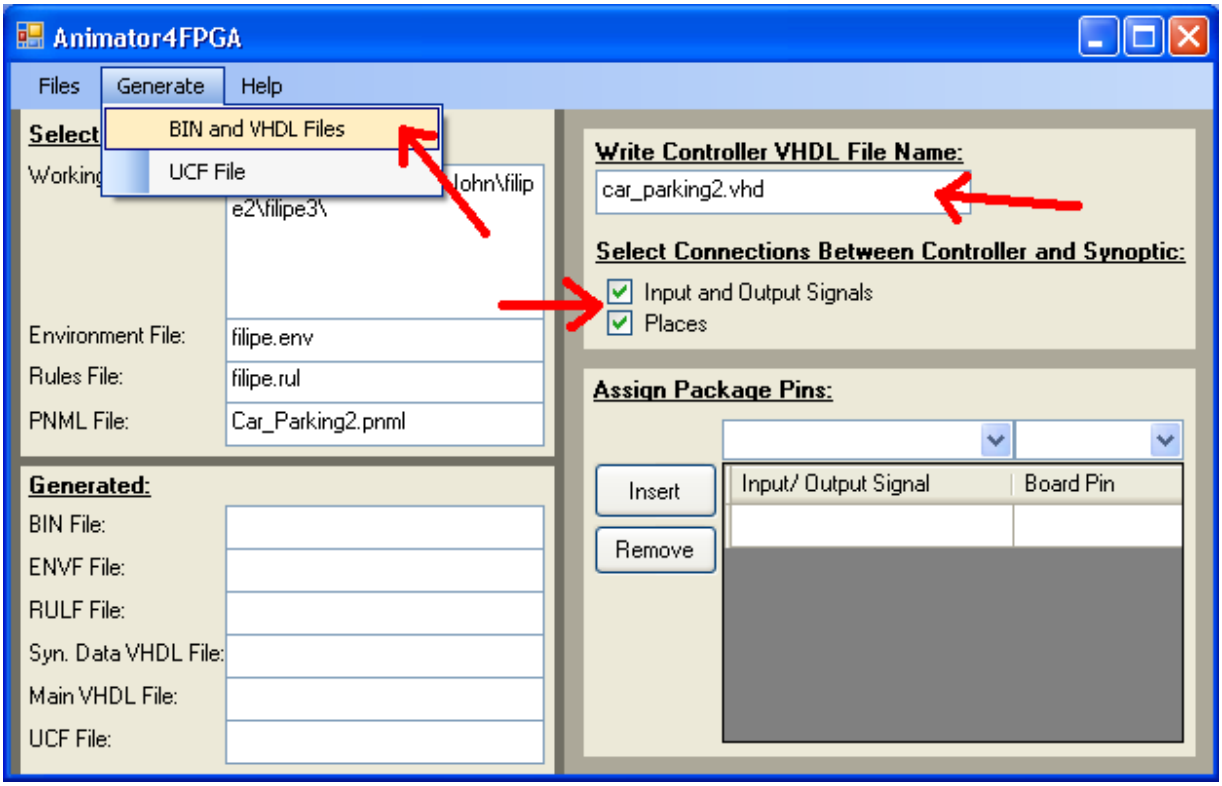


Figura 7.2. Geração dos ficheiros BIN e VHDL

Com esta acção são gerados 5 ficheiros:

- O ficheiro “BIN”,
- O ficheiro “ENVF” (é um ficheiro auxiliar),
- O ficheiro “RULF” (é também um ficheiro auxiliar),
- O ficheiro “Syn. Data VHDL File”,
- O ficheiro “Main VHDL File”.

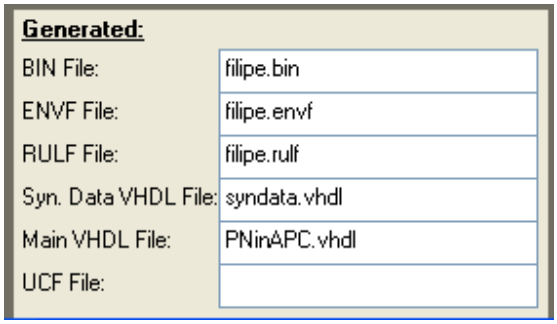


Figura 7.3. Ficheiros gerados

Para que estes ficheiros sejam gerados, é necessário que no momento de definição do ambiente e das regras no Animator sejam seguidas algumas condições em relação às imagens:

- As imagens a ser inseridas pelo Animator terão de estar em formato BMP,
- Todas as imagens usadas no Animator terão de ser previamente redimensionadas para a resolução final, dada a baixa resolução do LCD não seria viável redimensionar as imagens, pois em grande parte das situações iriam ficar deformadas. Assim o zoom definido no Animator será ignorado pelo Animator4Fpga, que ira processar as imagens como se tivessem todas um zoom de 100%,
- A imagem de fundo terá de ter uma resolução de 240x128,
- Nenhuma imagem poderá ter dimensões superiores a 240x128,
- Todas as imagens terão de ser totalmente inseridas dentro do fundo.

Ficheiro BIN

É o ficheiro binário que contém as imagens e que vai ser usado para configurar a EEPROM. Todas as são convertidas para preto e branco pelo Animator4Fpga, dado que o LCD utilizado só tem duas cores. Em cada byte contém a informação de 8 *pixels* de uma imagem.

Ficheiro Syn. Data VHDL File

Neste ficheiro está implementada a parte de dados do sinóptico. O ficheiro é gerado com base nos ficheiros RULF e ENVF, que por sua vez foram gerados com base nos ficheiros RUL e ENV. A parte de dados contém a seguinte informação:

- Posição onde estão armazenadas as imagens (a de fundo e as animadas) na EEPROM,

- Tamanho das imagens (a de fundo e as animadas),
- Largura das imagens (a de fundo e as animadas),
- Localização onde as imagens (a de fundo sempre na posição (0, 0), as animadas estáticas e as animadas com movimento) vão aparecer,
- Visibilidade das imagens (a de fundo é sempre visível, as animadas estáticas e as animadas com movimento, nestas a visibilidade varia de acordo com a localização),

Para além disto, esta parte de dados tem ainda a função de verificar os sinais e os lugares e indicar se cada regra está activa ou não.

Ficheiro Main VHDL File

Este ficheiro tem a implementação do *Main*, que faz as ligações entre a parte de dados do sinóptico, a parte de controlo do sinóptico e o controlador gerado pela ferramenta PNML2VHDL.

Geração do ficheiro UCF

Antes de gerar o ficheiro UCF é necessário atribuir os Pins da plataforma aos sinais de entrada e saída e aos eventos de entrada.

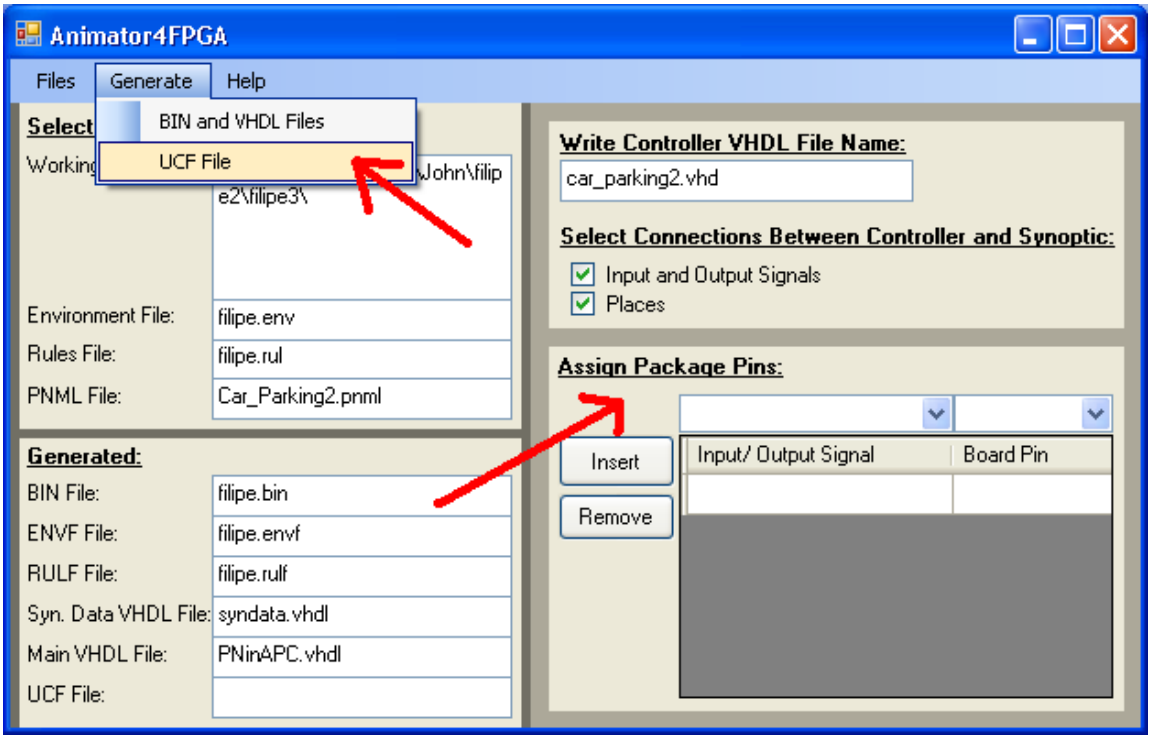


Figura 7.4. Geração do ficheiro UCF

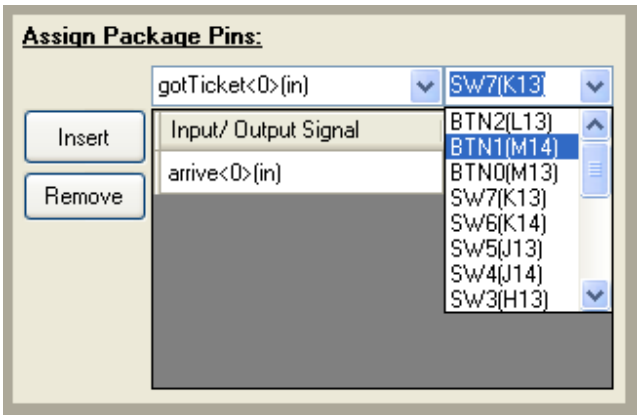


Figura 7.5. Atribuição de *Pins*

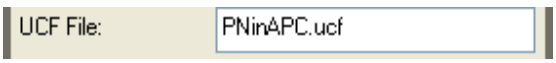


Figura 7.6. Ficheiro UCF gerado

Anexo B – Ficheiro ENVF

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!--APC Enviroment Structure-->
<Backgrounds>
  <Background>
    <Name>unnamed0</Name>
    <SourceStart>0</SourceStart>
    <SourceWidth>30</SourceWidth>
    <SourceLength>3840</SourceLength>
    <MonitorXLocation>0</MonitorXLocation>
    <MonitorYLocation>0</MonitorYLocation>
    <Animated>
      <Anim_Name>gate_down</Anim_Name>
      <Anim_SourceStart>3840</Anim_SourceStart>
      <Anim_SourceWidth>1</Anim_SourceWidth>
      <Anim_SourceLength>35</Anim_SourceLength>
      <Anim_BGLocation>1445</Anim_BGLocation>
      <Anim_Visible>True</Anim_Visible>
      <Anim_Transparency>False</Anim_Transparency>
      <Anim_Zoom>100</Anim_Zoom>
    </Animated>
    <Animated>
      <Anim_Name>gate_up</Anim_Name>
      <Anim_SourceStart>3875</Anim_SourceStart>
      <Anim_SourceWidth>5</Anim_SourceWidth>
      <Anim_SourceLength>20</Anim_SourceLength>
      <Anim_BGLocation>1445</Anim_BGLocation>
      <Anim_Visible>False</Anim_Visible>
      <Anim_Transparency>False</Anim_Transparency>
      <Anim_Zoom>100</Anim_Zoom>
    </Animated>
    <Animated>
      <Anim_Name>trafficlight_green</Anim_Name>
      <Anim_SourceStart>3895</Anim_SourceStart>
      <Anim_SourceWidth>3</Anim_SourceWidth>
      <Anim_SourceLength>24</Anim_SourceLength>
      <Anim_BGLocation>1142</Anim_BGLocation>
      <Anim_Visible>True</Anim_Visible>
      <Anim_Transparency>False</Anim_Transparency>
      <Anim_Zoom>100</Anim_Zoom>
    </Animated>
    <Animated>
      <Anim_Name>trafficlight_red</Anim_Name>
      <Anim_SourceStart>3919</Anim_SourceStart>
      <Anim_SourceWidth>3</Anim_SourceWidth>
      <Anim_SourceLength>24</Anim_SourceLength>
      <Anim_BGLocation>1142</Anim_BGLocation>
      <Anim_Visible>False</Anim_Visible>
      <Anim_Transparency>False</Anim_Transparency>
      <Anim_Zoom>100</Anim_Zoom>
    </Animated>
    <Animated>
      <Anim_Name>trafficlight_yellow</Anim_Name>
      <Anim_SourceStart>3943</Anim_SourceStart>
      <Anim_SourceWidth>3</Anim_SourceWidth>
      <Anim_SourceLength>24</Anim_SourceLength>
      <Anim_BGLocation>1142</Anim_BGLocation>
      <Anim_Visible>False</Anim_Visible>
      <Anim_Transparency>False</Anim_Transparency>
      <Anim_Zoom>100</Anim_Zoom>
    </Animated>
    <Animated>
```

```
      <Anim_Name>car1</Anim_Name>
      <Anim_SourceStart>3967</Anim_SourceStart>
      <Anim_SourceWidth>4</Anim_SourceWidth>
      <Anim_SourceLength>60</Anim_SourceLength>
      <Anim_BGLocation>1680</Anim_BGLocation>
      <Anim_Visible>True</Anim_Visible>
      <Anim_Transparency>False</Anim_Transparency>
      <Anim_Zoom>100</Anim_Zoom>
    </Animated>
    <Animated>
      <Anim_Name>maiordois</Anim_Name>
      <Anim_SourceStart>4027</Anim_SourceStart>
      <Anim_SourceWidth>4</Anim_SourceWidth>
      <Anim_SourceLength>84</Anim_SourceLength>
      <Anim_BGLocation>1636</Anim_BGLocation>
      <Anim_Visible>True</Anim_Visible>
      <Anim_Transparency>False</Anim_Transparency>
      <Anim_Zoom>100</Anim_Zoom>
    </Animated>
    <Animated>
      <Anim_Name>dois</Anim_Name>
      <Anim_SourceStart>4111</Anim_SourceStart>
      <Anim_SourceWidth>4</Anim_SourceWidth>
      <Anim_SourceLength>84</Anim_SourceLength>
      <Anim_BGLocation>1636</Anim_BGLocation>
      <Anim_Visible>True</Anim_Visible>
      <Anim_Transparency>False</Anim_Transparency>
      <Anim_Zoom>100</Anim_Zoom>
    </Animated>
    <Animated>
      <Anim_Name>um</Anim_Name>
      <Anim_SourceStart>4195</Anim_SourceStart>
      <Anim_SourceWidth>4</Anim_SourceWidth>
      <Anim_SourceLength>84</Anim_SourceLength>
      <Anim_BGLocation>1636</Anim_BGLocation>
      <Anim_Visible>True</Anim_Visible>
      <Anim_Transparency>False</Anim_Transparency>
      <Anim_Zoom>100</Anim_Zoom>
    </Animated>
    <Animated>
      <Anim_Name>zero</Anim_Name>
      <Anim_SourceStart>4279</Anim_SourceStart>
      <Anim_SourceWidth>4</Anim_SourceWidth>
      <Anim_SourceLength>84</Anim_SourceLength>
      <Anim_BGLocation>1636</Anim_BGLocation>
      <Anim_Visible>True</Anim_Visible>
      <Anim_Transparency>False</Anim_Transparency>
      <Anim_Zoom>100</Anim_Zoom>
    </Animated>
  </Background>
</Backgrounds>
```

Anexo C – Ficheiro RULF

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!--APC Rules-->
<Rules>
  <Rule>
    <RuleID>8</RuleID>
    <if>P2 = 1</if>
    <thenImgAddrStart>3967</thenImgAddrStart>
    <thenImgLength>60</thenImgLength>
    <thenImgWidth>4</thenImgWidth>
    <thenTransparency>0</thenTransparency>
    <thenLocation>1680</thenLocation>
    <thenZoom>100</thenZoom>
    <thenVisible>1</thenVisible>
    <thenTime>0</thenTime>
    <thenLocation>1681</thenLocation>
    <thenZoom>100</thenZoom>
    <thenVisible>1</thenVisible>
    <thenTime>1</thenTime>
  </Rule>
  <Rule>
    <RuleID>0</RuleID>
    <if>P4 = 0</if>
    <thenImgAddrStart>3919</thenImgAddrStart>
    <thenImgLength>24</thenImgLength>
    <thenImgWidth>3</thenImgWidth>
    <thenTransparency>0</thenTransparency>
    <thenLocation>1142</thenLocation>
    <thenZoom>100</thenZoom>
    <thenVisible>1</thenVisible>
    <thenTime>
    </thenTime>
  </Rule>
  <Rule>
    <RuleID>1</RuleID>
    <if>P4 = 1</if>
    <thenImgAddrStart>3943</thenImgAddrStart>
    <thenImgLength>24</thenImgLength>
    <thenImgWidth>3</thenImgWidth>
    <thenTransparency>0</thenTransparency>
    <thenLocation>1142</thenLocation>
    <thenZoom>100</thenZoom>
    <thenVisible>1</thenVisible>
    <thenTime>
    </thenTime>
  </Rule>
  <Rule>
    <RuleID>2</RuleID>
    <if>P4 &amp;gt; 1</if>
    <thenImgAddrStart>3895</thenImgAddrStart>
    <thenImgLength>24</thenImgLength>
    <thenImgWidth>3</thenImgWidth>
    <thenTransparency>0</thenTransparency>
    <thenLocation>1142</thenLocation>
    <thenZoom>100</thenZoom>
    <thenVisible>1</thenVisible>
    <thenTime>
    </thenTime>
  </Rule>
  <Rule>
    <RuleID>3</RuleID>
    <if>gateInOpen = 1</if>
    <thenImgAddrStart>3875</thenImgAddrStart>
    <thenImgLength>20</thenImgLength>
    <thenImgWidth>5</thenImgWidth>
    <thenTransparency>0</thenTransparency>
    <thenLocation>1445</thenLocation>
    <thenZoom>100</thenZoom>
    <thenVisible>1</thenVisible>
```

```
  <thenTime>
  </thenTime>
</Rule>
<Rule>
  <RuleID>4</RuleID>
  <if>gateInOpen = 0</if>
  <thenImgAddrStart>3840</thenImgAddrStart>
  <thenImgLength>35</thenImgLength>
  <thenImgWidth>1</thenImgWidth>
  <thenTransparency>0</thenTransparency>
  <thenLocation>1445</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>
  </thenTime>
</Rule>
<Rule>
  <RuleID>5</RuleID>
  <if>gateOutOpen = 1</if>
  <thenImgAddrStart>3875</thenImgAddrStart>
  <thenImgLength>20</thenImgLength>
  <thenImgWidth>5</thenImgWidth>
  <thenTransparency>0</thenTransparency>
  <thenLocation>1464</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>
  </thenTime>
</Rule>
<Rule>
  <RuleID>6</RuleID>
  <if>gateOutOpen = 0</if>
  <thenImgAddrStart>3840</thenImgAddrStart>
  <thenImgLength>35</thenImgLength>
  <thenImgWidth>1</thenImgWidth>
  <thenTransparency>0</thenTransparency>
  <thenLocation>1464</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>
  </thenTime>
</Rule>
<Rule>
  <RuleID>7</RuleID>
  <if>P1 = 1</if>
  <thenImgAddrStart>3967</thenImgAddrStart>
  <thenImgLength>60</thenImgLength>
  <thenImgWidth>4</thenImgWidth>
  <thenTransparency>0</thenTransparency>
  <thenLocation>1687</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>0</thenTime>
  <thenLocation>1689</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>1</thenTime>
  <thenLocation>1692</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>1</thenTime>
  <thenLocation>1696</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>1</thenTime>
  <thenLocation>1697</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>0</thenVisible>
  <thenTime>1</thenTime>
</Rule>
<Rule>
  <RuleID>9</RuleID>
  <if>P0 = 1</if>
```

```
  <thenImgAddrStart>3967</thenImgAddrStart>
  <thenImgLength>60</thenImgLength>
  <thenImgWidth>4</thenImgWidth>
  <thenTransparency>0</thenTransparency>
  <thenLocation>1682</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>0</thenTime>
  <thenLocation>1684</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>1</thenTime>
  <thenLocation>1686</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>1</thenTime>
</Rule>
<Rule>
  <RuleID>10</RuleID>
  <if>P6 = 1</if>
  <thenImgAddrStart>3967</thenImgAddrStart>
  <thenImgLength>60</thenImgLength>
  <thenImgWidth>4</thenImgWidth>
  <thenTransparency>0</thenTransparency>
  <thenLocation>1706</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>0</thenTime>
  <thenLocation>1706</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>0</thenVisible>
  <thenTime>1</thenTime>
</Rule>
<Rule>
  <RuleID>11</RuleID>
  <if>P7 = 1</if>
  <thenImgAddrStart>3967</thenImgAddrStart>
  <thenImgLength>60</thenImgLength>
  <thenImgWidth>4</thenImgWidth>
  <thenTransparency>0</thenTransparency>
  <thenLocation>1698</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>0</thenTime>
  <thenLocation>1699</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>1</thenTime>
  <thenLocation>1700</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>1</thenTime>
</Rule>
<Rule>
  <RuleID>12</RuleID>
  <if>P5 = 1</if>
  <thenImgAddrStart>3967</thenImgAddrStart>
  <thenImgLength>60</thenImgLength>
  <thenImgWidth>4</thenImgWidth>
  <thenTransparency>0</thenTransparency>
  <thenLocation>1701</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>0</thenTime>
  <thenLocation>1703</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>1</thenTime>
  <thenLocation>1705</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>1</thenTime>
</Rule>
```

```
 <Rule>
  <RuleID>13</RuleID>
  <if>P4 = 0</if>
  <thenImgAddrStart>4279</thenImgAddrStart>
  <thenImgLength>84</thenImgLength>
  <thenImgWidth>4</thenImgWidth>
  <thenTransparency>0</thenTransparency>
  <thenLocation>1636</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>
  </thenTime>
 </Rule>
 <Rule>
  <RuleID>14</RuleID>
  <if>P4 = 1</if>
  <thenImgAddrStart>4195</thenImgAddrStart>
  <thenImgLength>84</thenImgLength>
  <thenImgWidth>4</thenImgWidth>
  <thenTransparency>0</thenTransparency>
  <thenLocation>1636</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>
  </thenTime>
 </Rule>
 <Rule>
  <RuleID>15</RuleID>
  <if>P4 = 2</if>
  <thenImgAddrStart>4111</thenImgAddrStart>
  <thenImgLength>84</thenImgLength>
  <thenImgWidth>4</thenImgWidth>
  <thenTransparency>0</thenTransparency>
  <thenLocation>1636</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>
  </thenTime>
 </Rule>
 <Rule>
  <RuleID>16</RuleID>
  <if>P4 &amp;gt; 2</if>
  <thenImgAddrStart>4027</thenImgAddrStart>
  <thenImgLength>84</thenImgLength>
  <thenImgWidth>4</thenImgWidth>
  <thenTransparency>0</thenTransparency>
  <thenLocation>1636</thenLocation>
  <thenZoom>100</thenZoom>
  <thenVisible>1</thenVisible>
  <thenTime>
  </thenTime>
 </Rule>
</Rules>
```

Anexo D – Ficheiro Syn. Data

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity syndata is
Port (
    gotTicket : in std_logic_vector(0 downto 0);
    arrive : in std_logic_vector(0 downto 0);
    pay : in std_logic_vector(0 downto 0);
    leave : in std_logic_vector(0 downto 0);
    gateInOpen : in std_logic_vector(0 downto 0);
    gateOutOpen : in std_logic_vector(0 downto 0);
    P0 : in std_logic_vector(0 downto 0);
    P1 : in std_logic_vector(0 downto 0);
    P2 : in std_logic_vector(0 downto 0);
    P3 : in std_logic_vector(1 downto 0);
    P4 : in std_logic_vector(1 downto 0);
    P5 : in std_logic_vector(0 downto 0);
    P6 : in std_logic_vector(0 downto 0);
    P7 : in std_logic_vector(0 downto 0);
    ImgAddrStart : out std_logic_vector(12 downto 0);
    ImgLength : out std_logic_vector(11 downto 0);
    ImgWidth : out std_logic_vector(4 downto 0);
    BGAddrStart : out std_logic_vector(11 downto 0);
    ActiveRule : out std_logic;
    Visible : out std_logic;
    RulCountClear : in std_logic;
    RulCountInc : in std_logic;
    RulCountEnd : out std_logic;
    Time100ms : in std_logic;
    RESET : in std_logic;
    CLK : in std_logic
);
end syndata;

architecture Behavioral of syndata is
signal s_RulCountReg : std_logic_vector(4 downto 0);
signal s_ActiveRule0 : std_logic;
signal s_ActiveRule1 : std_logic;
signal s_ActiveRule2 : std_logic;
signal s_ActiveRule3 : std_logic;
signal s_ActiveRule4 : std_logic;
signal s_ActiveRule5 : std_logic;
signal s_ActiveRule6 : std_logic;
signal s_ActiveRule7 : std_logic;
signal s_ActiveRule8 : std_logic;
signal s_ActiveRule9 : std_logic;
signal s_ActiveRule10 : std_logic;
signal s_ActiveRule11 : std_logic;
signal s_ActiveRule12 : std_logic;
signal s_ActiveRule13 : std_logic;
signal s_ActiveRule14 : std_logic;
signal s_ActiveRule15 : std_logic;
signal s_ActiveRule16 : std_logic;
signal s_ActiveRule17 : std_logic;
signal s_BGAddrStartMov1 : std_logic_vector(11 downto 0);
signal s_VisibleMov1 : std_logic;
signal s_TimerRegMov1 : integer range 0 to 10;
signal s_BGAddrStartMov9 : std_logic_vector(11 downto 0);
signal s_VisibleMov9 : std_logic;
```

```
signal s_TimerRegMov9 : integer range 0 to 40;
signal s_BGAddrStartMov10 : std_logic_vector(11 downto 0);
signal s_VisibleMov10 : std_logic;
signal s_TimerRegMov10 : integer range 0 to 20;
signal s_BGAddrStartMov11 : std_logic_vector(11 downto 0);
signal s_VisibleMov11 : std_logic;
signal s_TimerRegMov11 : integer range 0 to 10;
signal s_BGAddrStartMov12 : std_logic_vector(11 downto 0);
signal s_VisibleMov12 : std_logic;
signal s_TimerRegMov12 : integer range 0 to 20;
signal s_BGAddrStartMov13 : std_logic_vector(11 downto 0);
signal s_VisibleMov13 : std_logic;
signal s_TimerRegMov13 : integer range 0 to 20;

begin

process(RESET, CLK)
begin
if RESET = '1' then
  s_RulCountReg <= "00000";
elsif CLK'event and CLK = '1' then
  if RulCountClear = '1' then
    s_RulCountReg <= "00000";
  elsif RulCountInc = '1' then
    s_RulCountReg <= s_RulCountReg + '1';
  end if;
end if;
end process;
RulCountEnd <= '1' when s_RulCountReg = "10001" else '0';

s_ActiveRule0 <= '1';
s_ActiveRule1 <= '1' when P2 = "1" else '0';
s_ActiveRule2 <= '1' when P4 = "00" else '0';
s_ActiveRule3 <= '1' when P4 = "01" else '0';
s_ActiveRule4 <= '1' when P4 > "01" else '0';
s_ActiveRule5 <= '1' when gateInOpen = "1" else '0';
s_ActiveRule6 <= '1' when gateInOpen = "0" else '0';
s_ActiveRule7 <= '1' when gateOutOpen = "1" else '0';
s_ActiveRule8 <= '1' when gateOutOpen = "0" else '0';
s_ActiveRule9 <= '1' when P1 = "1" else '0';
s_ActiveRule10 <= '1' when P0 = "1" else '0';
s_ActiveRule11 <= '1' when P6 = "1" else '0';
s_ActiveRule12 <= '1' when P7 = "1" else '0';
s_ActiveRule13 <= '1' when P5 = "1" else '0';
s_ActiveRule14 <= '1' when P4 = "00" else '0';
s_ActiveRule15 <= '1' when P4 = "01" else '0';
s_ActiveRule16 <= '1' when P4 = "10" else '0';
s_ActiveRule17 <= '1' when P4 > "10" else '0';

process(s_RulCountReg, s_ActiveRule0, s_ActiveRule1, s_ActiveRule2, s_ActiveRule3, s_ActiveRule4,
s_ActiveRule5, s_ActiveRule6, s_ActiveRule7, s_ActiveRule8, s_ActiveRule9, s_ActiveRule10, s_ActiveRule11,
s_ActiveRule12, s_ActiveRule13, s_ActiveRule14, s_ActiveRule15, s_ActiveRule16, s_ActiveRule17)
begin
  case s_RulCountReg is
    when "00000" => ActiveRule <= s_ActiveRule0;
    when "00001" => ActiveRule <= s_ActiveRule1;
    when "00010" => ActiveRule <= s_ActiveRule2;
    when "00011" => ActiveRule <= s_ActiveRule3;
    when "00100" => ActiveRule <= s_ActiveRule4;
    when "00101" => ActiveRule <= s_ActiveRule5;
    when "00110" => ActiveRule <= s_ActiveRule6;
    when "00111" => ActiveRule <= s_ActiveRule7;
    when "01000" => ActiveRule <= s_ActiveRule8;
    when "01001" => ActiveRule <= s_ActiveRule9;
    when "01010" => ActiveRule <= s_ActiveRule10;
```

```
    when "01011" => ActiveRule <= s_ActiveRule11;
    when "01100" => ActiveRule <= s_ActiveRule12;
    when "01101" => ActiveRule <= s_ActiveRule13;
    when "01110" => ActiveRule <= s_ActiveRule14;
    when "01111" => ActiveRule <= s_ActiveRule15;
    when "10000" => ActiveRule <= s_ActiveRule16;
    when "10001" => ActiveRule <= s_ActiveRule17;
    when others => ActiveRule <= '0';--0
end case;
end process;

process(s_RulCountReg)
begin
  case s_RulCountReg is
    when "00000" => ImgAddrStart <= "0000000000000";--0
    when "00001" => ImgAddrStart <= "0111101111111";--3967
    when "00010" => ImgAddrStart <= "0111101001111";--3919
    when "00011" => ImgAddrStart <= "0111101100111";--3943
    when "00100" => ImgAddrStart <= "0111100110111";--3895
    when "00101" => ImgAddrStart <= "0111100100011";--3875
    when "00110" => ImgAddrStart <= "0111100000000";--3840
    when "00111" => ImgAddrStart <= "0111100100011";--3875
    when "01000" => ImgAddrStart <= "0111100000000";--3840
    when "01001" => ImgAddrStart <= "0111101111111";--3967
    when "01010" => ImgAddrStart <= "0111101111111";--3967
    when "01011" => ImgAddrStart <= "0111101111111";--3967
    when "01100" => ImgAddrStart <= "0111101111111";--3967
    when "01101" => ImgAddrStart <= "0111101111111";--3967
    when "01110" => ImgAddrStart <= "1000010110111";--4279
    when "01111" => ImgAddrStart <= "1000001100011";--4195
    when "10000" => ImgAddrStart <= "1000000001111";--4111
    when "10001" => ImgAddrStart <= "0111110111011";--4027
    when others => ImgAddrStart <= "0000000000000";--0
end case;
end process;

process(s_RulCountReg)
begin
  case s_RulCountReg is
    when "00000" => ImgLength <= "111100000000";--3840
    when "00001" => ImgLength <= "000000111100";--60
    when "00010" => ImgLength <= "000000011000";--24
    when "00011" => ImgLength <= "000000011000";--24
    when "00100" => ImgLength <= "000000011000";--24
    when "00101" => ImgLength <= "000000010100";--20
    when "00110" => ImgLength <= "000000100011";--35
    when "00111" => ImgLength <= "000000010100";--20
    when "01000" => ImgLength <= "000000100011";--35
    when "01001" => ImgLength <= "000000111100";--60
    when "01010" => ImgLength <= "000000111100";--60
    when "01011" => ImgLength <= "000000111100";--60
    when "01100" => ImgLength <= "000000111100";--60
    when "01101" => ImgLength <= "000000111100";--60
    when "01110" => ImgLength <= "000001010100";--84
    when "01111" => ImgLength <= "000001010100";--84
    when "10000" => ImgLength <= "000001010100";--84
    when "10001" => ImgLength <= "000001010100";--84
    when others => ImgLength <= "000000000000";--0
end case;
end process;

process(s_RulCountReg)
begin
  case s_RulCountReg is
    when "00000" => ImgWidth <= "11110";--30
```

```
    when "00001" => ImgWidth <= "00100";--4
    when "00010" => ImgWidth <= "00011";--3
    when "00011" => ImgWidth <= "00011";--3
    when "00100" => ImgWidth <= "00011";--3
    when "00101" => ImgWidth <= "00101";--5
    when "00110" => ImgWidth <= "00001";--1
    when "00111" => ImgWidth <= "00101";--5
    when "01000" => ImgWidth <= "00001";--1
    when "01001" => ImgWidth <= "00100";--4
    when "01010" => ImgWidth <= "00100";--4
    when "01011" => ImgWidth <= "00100";--4
    when "01100" => ImgWidth <= "00100";--4
    when "01101" => ImgWidth <= "00100";--4
    when "01110" => ImgWidth <= "00100";--4
    when "01111" => ImgWidth <= "00100";--4
    when "10000" => ImgWidth <= "00100";--4
    when "10001" => ImgWidth <= "00100";--4
    when others => ImgWidth <= "00000";--0
end case;
end process;

process(s_RulCountReg, s_BGAddrStartMov1, s_BGAddrStartMov9, s_BGAddrStartMov10, s_BGAddrStartMov11,
s_BGAddrStartMov12, s_BGAddrStartMov13)
begin
  case s_RulCountReg is
    when "00000" => BGAddrStart <= "000000000000";--0
    when "00001" => BGAddrStart <= s_BGAddrStartMov1;--s_BGAddrStartMov1
    when "00010" => BGAddrStart <= "010001110110";--1142
    when "00011" => BGAddrStart <= "010001110110";--1142
    when "00100" => BGAddrStart <= "010001110110";--1142
    when "00101" => BGAddrStart <= "010110100101";--1445
    when "00110" => BGAddrStart <= "010110100101";--1445
    when "00111" => BGAddrStart <= "010110111000";--1464
    when "01000" => BGAddrStart <= "010110111000";--1464
    when "01001" => BGAddrStart <= s_BGAddrStartMov9;--s_BGAddrStartMov9
    when "01010" => BGAddrStart <= s_BGAddrStartMov10;--s_BGAddrStartMov10
    when "01011" => BGAddrStart <= s_BGAddrStartMov11;--s_BGAddrStartMov11
    when "01100" => BGAddrStart <= s_BGAddrStartMov12;--s_BGAddrStartMov12
    when "01101" => BGAddrStart <= s_BGAddrStartMov13;--s_BGAddrStartMov13
    when "01110" => BGAddrStart <= "011001100100";--1636
    when "01111" => BGAddrStart <= "011001100100";--1636
    when "10000" => BGAddrStart <= "011001100100";--1636
    when "10001" => BGAddrStart <= "011001100100";--1636
    when others => BGAddrStart <= "000000000000";--0
end case;
end process;

process(s_RulCountReg, s_VisibleMov1, s_VisibleMov9, s_VisibleMov10, s_VisibleMov11, s_VisibleMov12,
s_VisibleMov13)
begin
  case s_RulCountReg is
    when "00000" => Visible <= '1';--1
    when "00001" => Visible <= s_VisibleMov1;--s_VisibleMov1
    when "00010" => Visible <= '1';--1
    when "00011" => Visible <= '1';--1
    when "00100" => Visible <= '1';--1
    when "00101" => Visible <= '1';--1
    when "00110" => Visible <= '1';--1
    when "00111" => Visible <= '1';--1
    when "01000" => Visible <= '1';--1
    when "01001" => Visible <= s_VisibleMov9;--s_VisibleMov9
    when "01010" => Visible <= s_VisibleMov10;--s_VisibleMov10
    when "01011" => Visible <= s_VisibleMov11;--s_VisibleMov11
    when "01100" => Visible <= s_VisibleMov12;--s_VisibleMov12
    when "01101" => Visible <= s_VisibleMov13;--s_VisibleMov13
```

```
    when "01110" => Visible <= '1';--1
    when "01111" => Visible <= '1';--1
    when "10000" => Visible <= '1';--1
    when "10001" => Visible <= '1';--1
    when others => Visible <= '0';--0
end case;
end process;

process(RESET, CLK)
begin
if RESET = '1' then
  s_TimerRegMov1 <= 0;
elsif CLK'event and CLK = '1' then
  if Time100ms = '1' then
    if s_ActiveRule1 = '1' then
      if s_TimerRegMov1 < 10 then
        s_TimerRegMov1 <= s_TimerRegMov1 + 1;
      end if;
    else
      s_TimerRegMov1 <= 0;
    end if;
  end if;
end if;
end process;

process(s_TimerRegMov1)
begin
  if s_TimerRegMov1 >= 0 and s_TimerRegMov1 < 10 then
    s_BGAddrStartMov1 <= "011010010000";--1680
    s_VisibleMov1 <= '1';
  else
    s_BGAddrStartMov1 <= "011010010001";--1681
    s_VisibleMov1 <= '1';
  end if;
end process;

process(RESET, CLK)
begin
if RESET = '1' then
  s_TimerRegMov9 <= 0;
elsif CLK'event and CLK = '1' then
  if Time100ms = '1' then
    if s_ActiveRule9 = '1' then
      if s_TimerRegMov9 < 40 then
        s_TimerRegMov9 <= s_TimerRegMov9 + 1;
      end if;
    else
      s_TimerRegMov9 <= 0;
    end if;
  end if;
end if;
end process;

process(s_TimerRegMov9)
begin
  if s_TimerRegMov9 >= 0 and s_TimerRegMov9 < 10 then
    s_BGAddrStartMov9 <= "011010010111";--1687
    s_VisibleMov9 <= '1';
  elsif s_TimerRegMov9 >= 10 and s_TimerRegMov9 < 20 then
    s_BGAddrStartMov9 <= "011010011001";--1689
    s_VisibleMov9 <= '1';
  elsif s_TimerRegMov9 >= 20 and s_TimerRegMov9 < 30 then
    s_BGAddrStartMov9 <= "011010011100";--1692
    s_VisibleMov9 <= '1';
  elsif s_TimerRegMov9 >= 30 and s_TimerRegMov9 < 40 then
```

```
    s_BGAddrStartMov9 <= "011010100000";--1696
    s_VisibleMov9 <= '1';
  else
    s_BGAddrStartMov9 <= "011010100001";--1697
    s_VisibleMov9 <= '0';
  end if;
end process;

process(RESET, CLK)
begin
if RESET = '1' then
  s_TimerRegMov10 <= 0;
elsif CLK'event and CLK = '1' then
  if Time100ms = '1' then
    if s_ActiveRule10 = '1' then
      if s_TimerRegMov10 < 20 then
        s_TimerRegMov10 <= s_TimerRegMov10 + 1;
      end if;
    else
      s_TimerRegMov10 <= 0;
    end if;
  end if;
end if;
end process;

process(s_TimerRegMov10)
begin
  if s_TimerRegMov10 >= 0 and s_TimerRegMov10 < 10 then
    s_BGAddrStartMov10 <= "011010010010";--1682
    s_VisibleMov10 <= '1';
  elsif s_TimerRegMov10 >= 10 and s_TimerRegMov10 < 20 then
    s_BGAddrStartMov10 <= "011010010100";--1684
    s_VisibleMov10 <= '1';
  else
    s_BGAddrStartMov10 <= "011010010110";--1686
    s_VisibleMov10 <= '1';
  end if;
end process;

process(RESET, CLK)
begin
if RESET = '1' then
  s_TimerRegMov11 <= 0;
elsif CLK'event and CLK = '1' then
  if Time100ms = '1' then
    if s_ActiveRule11 = '1' then
      if s_TimerRegMov11 < 10 then
        s_TimerRegMov11 <= s_TimerRegMov11 + 1;
      end if;
    else
      s_TimerRegMov11 <= 0;
    end if;
  end if;
end if;
end process;

process(s_TimerRegMov11)
begin
  if s_TimerRegMov11 >= 0 and s_TimerRegMov11 < 10 then
    s_BGAddrStartMov11 <= "011010101010";--1706
    s_VisibleMov11 <= '1';
  else
    s_BGAddrStartMov11 <= "011010101010";--1706
    s_VisibleMov11 <= '0';
  end if;
```

```
end process;

process(RESET, CLK)
begin
if RESET = '1' then
  s_TimerRegMov12 <= 0;
elsif CLK'event and CLK = '1' then
  if Time100ms = '1' then
    if s_ActiveRule12 = '1' then
      if s_TimerRegMov12 < 20 then
        s_TimerRegMov12 <= s_TimerRegMov12 + 1;
      end if;
    else
      s_TimerRegMov12 <= 0;
    end if;
  end if;
end if;
end process;

process(s_TimerRegMov12)
begin
  if s_TimerRegMov12 >= 0 and s_TimerRegMov12 < 10 then
    s_BGAddrStartMov12 <= "011010100010";--1698
    s_VisibleMov12 <= '1';
  elsif s_TimerRegMov12 >= 10 and s_TimerRegMov12 < 20 then
    s_BGAddrStartMov12 <= "011010100011";--1699
    s_VisibleMov12 <= '1';
  else
    s_BGAddrStartMov12 <= "011010100100";--1700
    s_VisibleMov12 <= '1';
  end if;
end process;

process(RESET, CLK)
begin
if RESET = '1' then
  s_TimerRegMov13 <= 0;
elsif CLK'event and CLK = '1' then
  if Time100ms = '1' then
    if s_ActiveRule13 = '1' then
      if s_TimerRegMov13 < 20 then
        s_TimerRegMov13 <= s_TimerRegMov13 + 1;
      end if;
    else
      s_TimerRegMov13 <= 0;
    end if;
  end if;
end if;
end process;

process(s_TimerRegMov13)
begin
  if s_TimerRegMov13 >= 0 and s_TimerRegMov13 < 10 then
    s_BGAddrStartMov13 <= "011010100101";--1701
    s_VisibleMov13 <= '1';
  elsif s_TimerRegMov13 >= 10 and s_TimerRegMov13 < 20 then
    s_BGAddrStartMov13 <= "011010100111";--1703
    s_VisibleMov13 <= '1';
  else
    s_BGAddrStartMov13 <= "011010101001";--1705
    s_VisibleMov13 <= '1';
  end if;
end process;

end Behavioral;
```